Interactive deep generative models for symbolic music

Gaëtan Hadjeres

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Gaëtan HADJERES

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Modèles génératifs profonds pour la génération interactive de musique symbolique

soutenue le 7 juin 2018

devant le jury composé de :

M.	François Pachet	Directeur de thèse
M.	Frank Nielsen	Co-directeur de thèse
Mme.	Michèle Sebag	Rapporteur
M.	Marc Tommasi	Rapporteur
M.	Sylvain Marchand	Rapporteur
M.	Philippe Esling	Examinateur
M.	Douglas Eck	Examinateur

GAËTAN HADJERES

INTERACTIVE DEEP GENERATIVE MODELS FOR SYMBOLIC MUSIC

INTERACTIVE DEEP GENERATIVE MODELS FOR SYMBOLIC MUSIC

GAËTAN HADJERES
LIP6, Université Pierre et Marie Curie, Paris
Sony CSL, Paris

PhD thesis under the supervision of

FRANÇOIS PACHET
Sony CSL, Paris

and

FRANK NIELSEN
Sony CSL, Tokyo
École Polytechnique, Palaiseau

2014-2017

ABSTRACT

This thesis discusses the use of deep generative models for symbolic music generation. We will be focused on devising interactive generative models which are able to create new creative processes through a fruitful dialogue between a human composer and a computer.

Recent advances in artificial intelligence led to the development of powerful generative models able to generate musical content without the need of human intervention. I believe that this practice cannot be thriving in the future since the human experience and human appreciation are at the crux of the artistic production.

However, the need of both flexible and expressive tools which could enhance content creators' creativity is patent; the development and the potential of such novel A.I.-augmented computer music tools are promising.

In this manuscript, I propose novel architectures that are able to put artists back in the loop. The proposed models share the common characteristic that they are devised so that a user can control the generated musical contents in a creative way. In order to create a user-friendly interaction with these interactive deep generative models, user interfaces were developed. I believe that new compositional paradigms will emerge from the possibilities offered by these enhanced controls. This thesis ends on the presentation of genuine musical projects like concerts featuring these new creative tools.

RÉSUMÉ

Ce mémoire traite des modèles génératifs profonds appliqués à la génération automatique de musique symbolique. Nous nous attacherons tout parliculièrement à concevoir des modèles génératifs interactifs, c'est-à-dire des modèles instaurant un dialogue entre un compositeur humain et la machine au cours du processus créatif.

En effet, les récentes avancées en intelligence artificielle permettent maintenant de concevoir de puissants modèles génératifs capables de générer du contenu musical sans intervention humaine. Il me semble cependant que cette approche est stérile pour la production artistique dans le sens où l'intervention et l'appréciation humaines en sont des pilliers essentiels.

En revanche, la conception d'assistants puissants, flexibles et expressifs destinés aux créateurs de contenus musicaux me semble pleine de sens. Que ce soit dans un but pédagogique ou afin de stimuler

la créativité artistique, le développement et le potentiel de ces nouveaux outils de composition assistée (ou augmentée) par ordinateur sont prometteurs.

Dans ce manuscrit, je propose plusieurs nouvelles architectures remettant l'humain au centre de la création musicale. Les modèles proposés ont en commun la nécessité de permettre à un opérateur de contrôler les contenus générés. Afin de rendre cette interaction aisée, des interfaces utilisateurs ont été développées ; les possibilités de contrôle se manifestent sous des aspects variés et laissent entrevoir de nouveaux paradigmes compositionnels. Afin d'ancrer ces avancées dans une pratique musicale réelle, je conclue cette thèse sur la présentation de quelques réalisations concrètes (partitions, concerts) résultant de l'utilisation de ces nouveaux outils.

PUBLICATIONS

[1] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. "Deep Learning Techniques for Music Generation-A Survey." In: *arXiv preprint arXiv:1709.01620* (2017).

[2] Gaëtan Hadjeres and Frank Nielsen. "Deep rank-based transposition-invariant distances on musical sequences." In: *arXiv preprint arXiv:1709.00740* (2017).

[3] Gaëtan Hadjeres and Frank Nielsen. "Interactive Music Generation with Positional Constraints using Anticipation-RNNs." In: *arXiv preprint arXiv:1709.06404* (2017).

[4] Gaëtan Hadjeres, Frank Nielsen, and François Pachet. "GLSR-VAE: Geodesic Latent Space Regularization for Variational AutoEncoder Architectures." In: *arXiv preprint arXiv:1707.04588* (2017). Accepted to SSCI 2017.

[5] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. "DeepBach: a Steerable Model for Bach Chorales Generation." In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2017, pp. 1362–1371. URL: http://proceedings.mlr.press/v70/hadjeres17a.html.

[6] Gaëtan Hadjeres, Jason Sakellariou, and François Pachet. "Style Imitation and Chord Invention in Polyphonic Music with Exponential Families." In: *CoRR* abs/1609.05152 (2016). URL: http://arxiv.org/abs/1609.05152.

[7] Frank Nielsen and Gaëtan Hadjeres. "Approximating Covering and Minimum Enclosing Balls in Hyperbolic Geometry." In: *Geometric Science of Information - Second International Conference, GSI 2015, Palaiseau, France, October 28-30, 2015, Proceedings*. 2015, pp. 586–594.

[8] Frank Nielsen and Gaëtan Hadjeres. "Monte Carlo Information Geometry: The dually flat case." In: *CoRR* abs/1803.07225 (2018). arXiv: 1803.07225. URL: http://arxiv.org/abs/1803.07225.

[9] Bob L. Sturm, Oded Ben-Tal, Ùna Monaghan, Nick Collins, Dorien Herremans, Elaine Chew, Gaëtan Hadjeres, Emmanuel Deruty, and Françcois Pachet. *Machine Learning in Music Practice: A Case Study*. Submitted to Journal of New Music Research.

À toi, Kypris victorieuse,
je consacre ces offrandes encore mouillées de rosée,
vestiges des douleurs de la vierge,
témoins de mon sommeil et de ma résistance.

— Pierre Louÿs, *Le sommeil interrompu*
in *Les chansons de Bilitis*

ACKNOWLEDGMENTS

I would like to thank my Ph. D. supervisor François Pachet for having introduced me to the fascinating problem of automatic music generation. I am particularly grateful for the trust he put in me and for the unprecedented freedom I had during these three years in the Sony Computer Science Laboratory under his direction. Working in such a fruitful research environment on a topic that interests me more and more as days go by is an immense privilege for which I am all the more beholden to him.

A very special gratitude goes to my thesis co-director, Frank Nielsen, whose dedication and involvement I cannot but praise. He transmitted to me his keen interest in information geometry and guided me throughout the many hardships one can encounter during their Ph.D.

I want to sincerely thank all the members of my jury: Michèle Sebag, Sylvain Marchand, Marc Tommasi, Philippe Esling and Douglas Eck for having accepted to devote their time to evaluate and comment my research. A special thanks is due to the reviewers of my thesis: Michèle Sebag, Sylvain Marchand and Marc Tommasi. I want to thank them for having carefully read my manuscript and provided me with insightful comments. Their interest in my work and their encouragement gave me the will to continue to delve into this research topic.

I also thank the École Polytechnique for the stimulating education I received during three years and for the Ph. D. grant I have been awarded. This helped me discover new research horizons and it had a considerable impact on my life.

A particularly warm thanks goes to my family: my mother, my brother and my girlfriend, for their unwavering support and unfailing trust.

Lastly, I thank all the people who supported me (or bore me) during these passionate but demanding years and whose name I will not include here in order not to be too personal nor too lengthy. This includes all the people I collaborated with, my former colleagues at Sony CSL and my longtime friends. Their company was a crucial element in the completion of this thesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

MOTIVATIONS

The introduction of deep learning techniques into the scope of generative modeling has led to the development of powerful and expressive generative models. Recent advances showed for instance that it is possible to generate captions from images [124, 154], images from captions [101, 129, 130] or audio from text [55, 114]. Interestingly enough, these techniques are not easily transferable into the domain of symbolic music generation i.e. generation of music sheets. Indeed, music has its own structure and its own rules and it is intrinsically different from all the other types of data. Devising generative models for symbolic music gives rise to many novel and valuable computational problems.

A natural question which is often asked when speaking about the automatic generation of music is the following: What for? Why would we give up our artistic faculties to computers and become mere operators?

Composing music is indeed profoundly considered as an artistic process where artificial intelligence would have no role to play in. Worst, we have a strong reluctance in appreciating a music which has not been composed by a human and I think it has to be so. We can appreciate the quality of a computer-generated piece, but this work will never be considered as a piece of art. Art is more about communicating human thought processes and human experiences than about the resulting product. However, many recent deep generative models on music tend to produce black box models only able to spit out an infinite number of musical pieces. There is no human intervention, except when building up the model. If they participate to the proposing of challenging problems in artificial intelligence, these models cannot be used to actually compose music and can only be seen as a proof of concept. This criticism has been formulated before, but not in the context of deep generative models. This resulted in models with which we could play with, but which did not have the modeling and generalization capabilities of deep learning models.

Can we take the best of both worlds? Is it possible to put the human back in the loop by devising deep generative models that would enhance composers creativity rather than putting them aside?

We will explore in this manuscript how to build upon the recent and outstanding progresses in artificial intelligence in order to serve artistic purposes. Since I am convinced that humans must intervene

in any artistic process, our focus will then be on *interactive deep generative models for symbolic music generation*; that is, devising deep generative models that users could interact with in order to help them during composition. This theme, which has not been previously stated as such before, is at the intersection between many domains and I hope this thesis will participate in its acceptance and in its development.

I strongly believe that the requirement for a generative model of music to be interactive is essential, since it anchors the related computational problems onto the range of real-world problems. We must not only design powerful generative methods but also need to rethink about the human-machine interaction itself. This creates both concrete and theoretical problems. Indeed, when working on interactive deep generative models, the theoretical machinery can only express itself through the development of appropriate user interfaces. This raises original and numerous questions like:

- What features are desirable for a computer-assisted composition?
- How can we boost users' creativity or users' productivity?
- Which kind of interaction do we need and what interface it implies?
- Can novelties in deep generative models suggest new compositional processes?
- How to design both powerful and flexible deep generative models fulfilling well-specified musical and technical requirements?

Many kind of interactions can be thought about, and each one requires the design of novel approaches. Musical ideas can lead to the development of new A.I. models and vice-versa. Not only this interaction is fruitful, but it can also help to have a better understanding of how these fascinating deep generative models behave.

CONTRIBUTIONS

This thesis is composed of three parts.

Part I consists in an overview on deep generative techniques for music composition. The focus will be on exposing general ideas on symbolic music generation and how a careful look upon the musical data can help in designing better models. This part starts by first presenting the available symbolic musical data in Chap. 2. In this chapter, we discuss about the various formats in which we can find musical data and highlight what are the most important and unique features in music that we have to bear in mind when building generative models. It finally ends with a look upon the different datasets commonly

used for automatic music generation. A particular attention will be paid to the dataset of the chorale harmonizations by J.S. Bach as it will be used in many of the experiments presented in this work. Chapter 3 deals with the particular issues and challenges encountered when building generative models for symbolic music. Namely, we discuss about the importance of crafting appropriate data representations depending on the musical data at stake and about the modeling issues caused by the lack of an objective evaluation on the generated pieces. We then conclude by emphasizing upon the importance of rethinking the interaction between a user and the machine when designing generative models. The following chapter, Chap. 4, is a non exhaustive survey on the recent approaches on deep generative models for music. For each, we will present its drawbacks and advantages with respect of the issues listed in the preceding chapter. We will see that most of the mentioned methods, despite their relatively expressive power, cannot be used in an interactive way which is, in my opinion, a severe limitation; one cannot really "play" or compose with them.

Part II exposes my work on polyphonic music generation. It is split into two chapters which deal with the same problem: learning and generating chorales in the style of J.S. Bach. Chapter 5 must be perceived as an introduction to its succeeding chapter. In it, I expose and analyze a solution which is not built using deep learning techniques. The advantage of this solution is its clear mathematical formulation together with a novel method to handle polyphonic music. The proposed model is able to be used interactively and its invention capacities are evaluated. By identifying and correcting the weaknesses of this model, I have developed DeepBach, a major improvement over the preceding model. This model, presented in Chap. 6, is able to generate musically-convincing chorales in the style of Bach while being steerable. This means that composition can now be thought of as a constructive dialogue between a composer and the model. In order to make this new interactive compositional process be smooth, we conceived a graphical interface so that a user can naturally query the system. With it, even non experts can compose well-written complex chorale music. Many of DeepBach generated chorales have been played live and the music sheets are available in Appendix A.

In part III, I present independent techniques in the simpler case of monophonic sequence generation. These techniques are different approaches to the same problem: thinking about new ways of interacting with a generative model. For instance, having a perceptually "good" distance between music sequences is of great use in music generation since we could use it to generate variations upon a given pattern. This is considered in Chap. 7 where I propose a method to create a transposition-invariant distance. This allows to detect patterns and their variations irrespective of the key they are in. Contrary to other existing distances on sequences, this distance is learned

from data and thus better matches our perception of proximity in the space of sequences. Chapter 8 is a short chapter exposing an original method to generate sequences with unary constraints while sampling sequences with the correct probabilities. Adding unary constraints is a way to explore the capabilities of a generative model and a first step towards the instauration of a dialogue between a composer and the machine. The great advantage of this method is its efficiency at generation time. Having such a fast generation scheme is all the more interesting in the context of real-time interactive applications for instance. In the last chapter, Chap. 9, I propose to enrich a class of powerful generative models by adding meaningful ways to control its generations. The most interesting feature here is that this control is continuous. This method is applicable not only in music, but in almost every domain, from image generation to molecule synthesis. In the context of sequence generation, this allows to smoothly create variations of a given melody in an intended way. Since generation is instantaneous and the possibilities of control infinite, I believe this model can be used to propose an elegant, novel and intuitive way of composing music.

Part I

OVERVIEW

2

MUSICAL SYMBOLIC DATA

With the advent of big data analytics, data is more and more valuable and the amount of available data grows at an unprecedented pace. This is true in nearly every domain... except for the one we are interested in. Indeed, musical symbolic data is really scarce: contrary to text or images, it is "hard" to produce by individuals; most of the musical contents nowadays is not in a symbolic format and the automated production of musical symbolic data designed for learning is nonsensical.

So, why bother with a symbolic view and concentrate directly on the raw audio which is the true nature of music?

We think that considering only musical notations helps reducing the variability of the datasets (all possible interpretations) and focus on the *essence* or the *idealized* view of the music. The music notation is always an abstraction: it is a way to represent a musical performance. There are numerous musical notation systems, each one adapted for notating a particular type of music, some are elusive while others can be very precise.

A musical notation is *adapted* when its users (composers, performers) are able to communicate their musical intentions in a way that suits their artistic desires with minimum effort.

In the same way, a musical data representation in automatic music generation will be *adapted* if a generative model is able to easily learn how to generate convincing musical contents. Hence, a representation must not be too complex or detailed (so that the model focuses on the "most important" aspects of the music) nor too simple (otherwise the model could skip important features of the music). Finding the correct "granularity" is all the more essential in music generation because of the lack of huge corpora.

The aim of this chapter is to expose the singularity of the musical symbolic data and show that it is an endless reservoir of new computational challenges.

This chapter starts in Sect. 2.1 by introducing the main formats in which music is encoded. We describe in Sect. 2.2 the specificities and the peculiarities of the musical data and how it differs drastically from other (apparently close) data such as textual data. With these highlights in mind, we expose in Sect 2.3 the available datasets commonly used in musical symbolic learning.

SYMBOLIC MUSIC NOTATION FORMATS

The modern Western musical notation format

When talking about musical notation, we immediately think about the *modern Western musical notation* (Fig. 1). This sophisticated notation include notes and rhythms and can include tempo indications, dynamics, expressive annotations, musical phrases, lyrics, etc. so that composers can express and transmit their musical ideas. This notation is both precise and elusive.

Precise since every played note is written on the score (in most of the occidental music, there is few place left for improvisation); elusive because the dynamics or the musical phrases are to be interpreted. The player's understanding of the piece and the music at stake is thus essential for a decent performance.

In fact, it is more than precise since it also contains a lot of implicit expert musical knowledge. A simple example of this is the presence of the key signature (in Fig 1, this is represented by the first five flats right after the clefs which denote that every time one of these five pitches is found on the music sheet, it must be played flattened). It does not only provide a convenient way to remove from the score too many accidentals but conveys a musical meaning: the same music sheet but with another key signature would appear misspelled. There are a lot of historical conventions in occidental classical music: this format is thus not only a way of notating notes but also bears an additional meaning. This is also true in the spelling of the notes: in a word, the G♭ of mes.4 in Fig. 1 could not be written F♯ even if these two notations would sound the same.

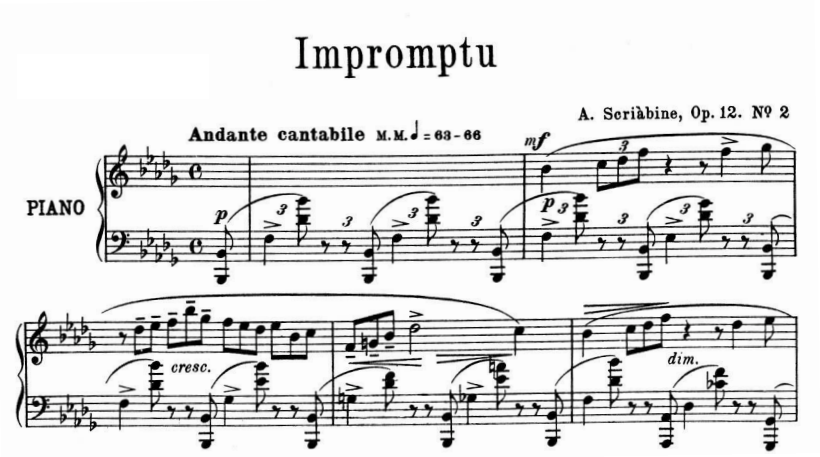


Figure 1: Example of modern Western musical notation: Impromptu Op. 12 n°2 by Scriabin (1895).

This notation with five-line staves is used by musicians of many different genres all over the world since centuries. An interesting spe-

cialized use of this notation is the lead sheet notation, which consists of a staff of chord symbols together with a staff containing a melody (Fig. 2). This notation is primarily used for notating pop songs and jazz standards. It is more evasive than the classical music sheets. A performer is more free in their musical choices and the lead sheet is more an abstract guide than a high-fidelity description of a performance: the musical ensemble is not specified; the chords are only in a symbolic format; the tempo indication is vague; etc. It is a way to fix some ideas while letting some other parameters free.

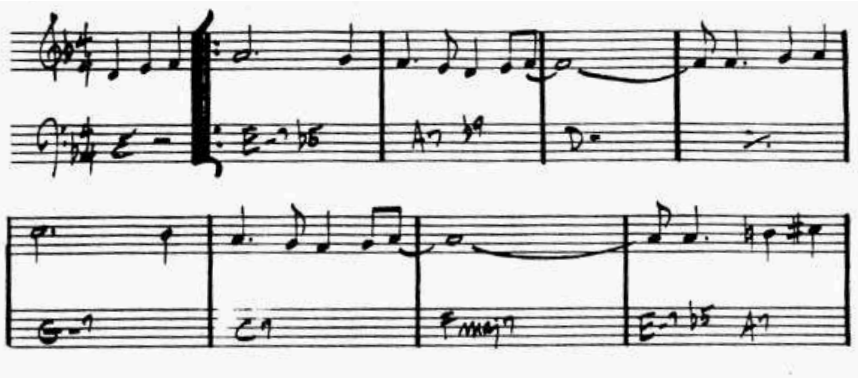


Figure 2: Example of a lead sheet: first measures of Beautiful Love by Wayne King, Victor Young and Egbert Van Alstyne (1931).

However, this purely graphical notation may not be suitable from a computational point of view. The music sheet of Fig. 1 contains in fact far more information than the ones we already mentioned: from the stems direction to the precise location of the dynamics. An accurate representation, not in image format, of this notation must translate all this complexity.

Markup languages

A first step towards a computer-readable format is the MusicXML open format (see Fig. 3). It is a markup language providing support for notating all "important" musical information (key, time signatures, clefs, beaming information, stem directions, slurs, ornaments, bar lines etc.). If this format is of great use for sharing musical files, this representation is not suitable from a learning perspective since it includes a lot of overhead.

Other related formats used to describe musical objects can be devised, each one with its specificity. These formats are not easily readable by a human and need a music notation software to easily work with. The advantage of the MusicXML format is that it can be read by most of the score writing programs and focuses primarily on the "essence" of the music sheet, while other proprietary formats (used

```
<note>
  <pitch>
    <step>C</step>
    <octave>4</octave>
  </pitch>
  <duration>4</duration>
  <type>whole</type>
</note>
```

Figure 3: Example of a single note in the MusicXML format.

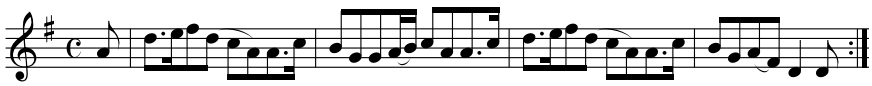
for instance by *Finale* or *Sibelius* notation software) also include graphical and formatting information.

ABC notation

The traditional Western music notation system is the most widely used one, but other musical notation exists. One example is the ABC notation [157], a simple textual notation for notating simpler musical pieces like monophonic melodies. Its simplicity and its readability (compared to the MusicXML format) makes it widely useful for notating and sharing folk and traditional tunes of Western Europe.

```
T:All The Way To Galway
M:C
L:1/8
K:Dmix
A|d>ef(d cA)A>c|BGG(A/B/) cAA>c|
d>ef(d cA)A>c|BG(AF) D2D:|
```

(a) ABC format.



(b) Rendering in traditional music notation.

Figure 4: Example of a monophonic melody in ABC format and its rendering in Western musical notation: All The Way To Galway, anon., Ireland, from Francis O'Neill's "The Dance Music of Ireland" (1907).

This notation is primarily meant for transcribing monophonic melodies: it focuses on notating pitch and rhythm. Other information are available in the preamble of the document (such as the title, the key signature or the default length of a note). Without entering into the details of the notation, the name of a note is notated using its corresponding English notation (from A to G), its accidental is notated by ^ (sharp) and _ (flat) and its duration is transcribed by an integer or a fraction following the note name. This number indicates the du-

ration of the note as a multiplier of the default note length. Using lowercase or uppercase letters help differentiate the different octaves. Other symbols and several shortcuts exist in order to make this notation rich enough and as light as possible (see Fig. 4 for an example).

This notation also includes the possibility to write chord symbols or multiple-note chords, but these features have some limitations.

MIDI

Even if it is not strictly speaking a symbolic music format, the MIDI format can be used as such. It is in fact designed to record live performances rather than symbolic data.

MIDI (Musical Instrument Digital Interface) [96] carries *event messages* that specify note information such as its *pitch* and the *velocity* with which it has been played. There are five types of message and here we only consider/describe the *Channel Voice* type, which transmits real-time performance data over a single channel.

For a given channel, these messages are tuples of the type

(Note on/off, MIDI pitch, velocity),

where the *MIDI pitch* is an integer between 0 and 127, where each integer corresponds to a unique frequency. As an example, the middle C is encoded as 60 and the C♯/D♭ frequency as 61. With this notation, it is interesting to note that there is no way to differentiate between the different *spellings* of *enharmonic* notes (two notes that sound the same but are written differently). The velocity is also an integer between 0 and 127.

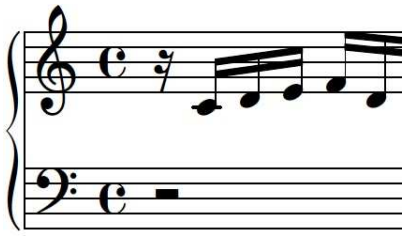
The *note on* event indicates that a note is being played from then on (e.g. a key has been pressed) and the corresponding *note off* event indicates that this note is no longer being played (e.g. the key has been released).

These messages can be used to transmit real-time performances or can be attributed a time value and stored in a file in order to save performances. These time values must be integers and represents the number of ticks (minimal duration between two events) since the beginning of the piece.

MIDI files can be used for instance to transcribe music sheets (written in the modern music notation format) by saving idealized (quantized) performances. An example of extract from a MIDI file (turned into readable ASCII) and its corresponding score is shown in Fig. 5. In this example, there are 384 ticks per quarter note, which corresponds to 96 ticks for an eighteenth note, and all notes have the same velocity. It is possible to play multiple notes at the same time (polyphony) within the same channel.

```
 96,  Note_on_c, 0, 60, 90
192, Note_off_c, 0, 60, 0
192,  Note_on_c, 0, 62, 90
288, Note_off_c, 0, 62, 0
288,  Note_on_c, 0, 64, 90
384, Note_off_c, 0, 64, 0
384,  Note_on_c, 0, 65, 90
480, Note_off_c, 0, 65, 0
480,  Note_on_c, 0, 62, 90
576, Note_off_c, 0, 62, 0
```

(a)



(b)

Figure 5: Extract from a (simplified) MIDI file (a) and its corresponding score (b).

SINGULARITIES OF THE SYMBOLIC MUSICAL DATA.

In this section, we focus on the specificities of the symbolic musical data and why it differs drastically from other types of data commonly used in machine learning such as text or images. Consider as an illustration of this claim the following question: what makes monophonic melodies in ABC format representation require a distinct treatment from the ones used for natural language processing?

Melody, harmony and rhythm

Most of the music we hear is strongly rooted in the Western music tradition or common-practice period and we will from then on only focus on this kind of music. Whether it is in jazz music or in traditional Irish tunes, these music have in common the notion of melody, harmony and rhythm, inherited from the past. If the terms melody and rhythm are easily understandable, the notion of harmony can be less clear. The term harmony roughly designates the principles behind the simultaneous superposition of sounds (chords) and how these chords are arranged into chord progressions. In this sense, it is a "vertical" view (when written on a music sheet) of the music. This is opposed to the melodic or "horizontal" view of the music i.e. how a single voice evolves through time. Generally speaking, chords are composed of three to five notes and may be rearranged as a stack of thirds. This stems from physical principles coupled with the long and progressive evolution of the occidental music.

When playing a given chord, playing a note belonging to that chord at the same time will sound *consonant* while playing non-chord note at the same time will sound *dissonant*. Dissonance is not a thing to avoid in classical music, on the contrary, notes perceived as expressive are often dissonant. However these notes must be carefully arranged so that they resolve on consonant notes. Roughly speaking, we can

say that music as we know it is based on the creation followed by the resolution of musical tensions.

It is important to pinpoint that, even when there is only one voice in a piece, this notion of chord is still present. For instance, in Fig. 4, the underlying harmony is immediately perceivable: the most important notes of the melody are notes composing the implied chords.

When speaking of rhythm, we do not only talk about the timing between two consecutive notes, but also include the notion of meter of the music. All the music we will consider is regularly-pulsed, but the hierarchy between these evenly-spaced pulses can vary. This is of great importance since it is at the crux of the differences between dances for instance. This hierarchy between between pulses or beats repeats itself through time which allows the perception of higher-level groups called measures or bars. As an example, a waltz will be composed of three-beat measures whose first beat will be more important than the two following beats. The frequency of the harmonic changes is called the *harmonic rhythm* and must be in accordance with the hierarchy imposed by the meter. Changes cannot occur too swiftly in order to make the harmony (or implied harmony) audible. In examples of Fig. 1 and 4, the harmonic changes occur every two beats in a four-beat measures.

To put it in a nutshell, even in the simple melody of Fig. 4, the choice of the pitches and their organization through time relies on our subconscious understanding of these musical rules inherited from our perpetual exposition to music. This is why we are able to detect "wrong notes" or "bad timings", even if we have no musical education.

Structure, motives and patterns

Music has its own organization. Contrary to text, music loves to repeat itself. Indeed, most of the songs we know alternate between verses and a refrain. Basically, the refrain is repeated identically at each of its occurrences while the verses have the same music but different lyrics.

There are many reasons why such a structure is so popular. The aim when composing music is to attract and hold the interest of a listener. The alternation between two contrasting parts is a way to achieve this goal: the verses renew the interest in the music and prevents the listener from getting bored, while the refrain intervenes as a landmark, which gives coherence.

The high-level structure of a musical piece is thus a way to construct long, but coherent, pieces. We will not delve into the details of the many possible structures of musical pieces: it is the result of a long history of musical practice [2]. Furthermore, we believe that this notion is far from being captured by generative models.

Music is also structured on a more local scale. Indeed, in order to give coherence to a particular piece, some *motives* or *patterns* are introduced and used throughout the musical piece. All musical examples presented so far constitute interesting examples of such a motivic treatment. In Fig. 1, a characteristic rhythm coupled with an ascending motif is introduced by the left hand at the beginning of the piece. This pattern is then repeated during all the following bars; it can be modified at some points in order to follow the underlying harmonies but is still recognizable as a variation of the initial pattern. In Fig. 2, the first four bars introduce a harmonized melody in the key of D minor. This musical phrase is then transposed a third higher in the key of F major. The subtlety here is that such a transformation seems "natural": we can anticipate what the following notes will be given the initial motif and the first notes of its successor. However, there are in fact some changes between both like a slight change of rhythm or the fact that it is not an exact transposition: the melody or motif has been adapted to a new harmonization. The example of Fig. 4 follows another organization principle: a first musical phrase of two bars (named *antecedent phrase*) is followed by a second two-bar musical phrase (named *consequent phrase*). The beginning of these two phrases is the same while they have different endings and different cadences (chord progressions that convey a sense of resolution or suspension).

In all cases, these repetitions and transformations create a sequence of fulfilled/unfulfilled expectations on an attentive listener and contribute to the overall continuity and coherence of the musical pieces. All the art (or craftsmanship) in music composition consists in the careful and thriving interplay between musical phrases and multiple-scale rhythmic changes, harmonic changes and motivic transformations.

Style

Until now, we have not mentioned the notion of style in music but dwelled on general principles shared among varied musical styles and epochs. Even if it is an ill-defined notion, it is nonetheless the most recognizable feature which helps us distinguish, for instance, Bach from Debussy. Which chords are used, how they are played (*voicing*), how pieces are structured and how melodic lines interact with the harmonic skeletons are characteristic elements of each composer and evolved through history. Despite the shared characteristics between musical pieces produced in two different epochs, the notion of style appears to be crucial in our appreciation of music: transposing formulations found in Chopin's works to Mozart's compositions would simply appear "out of style" and even discordant. These works

have been composed with totally different aims and different compositional processes.

The style of a musical piece has an inarguable impact on its appreciation. Therefore, a good generative model should be able to differentiate clearly between different styles and epochs in order to be able to generate musically-convincing works.

For these reasons, I believe that designing such a generative model that would encompass all western music is for the moment out of reach. In order to generate plausible music that we can evaluate from a musical standpoint, I suggest that we must focus on problems of the form: "Given a dataset in the style of X, generate a new musical content in the same style".

SYMBOLIC MUSIC DATASETS

We now quickly describe some of the most important datasets available for automatic symbolic music composition. Each of these datasets has its own characteristics (format, instrumentation, musical genre, etc.) and requires a specific attention. A common feature is their relatively small size compared, for instance, to image or text datasets (ImagetNet [42, 133] is composed of nearly 15 million images while the One Billion Word dataset [24] contains, as its name indicates, almost one billion words of training data).

Monophonic datasets

Monophonic datasets in symbolic music are often datasets of traditional or popular music.

Folk Songs

The *Nottingham dataset*[1] is a collection of 1200 British and American folk tunes. It is composed of tunes in the ABC format and includes chord labels. Its conversion in the MIDI format is often used as a baseline in music. However, this conversion makes little sense: the choice on how to actually play the chord symbols is always the same and contrary to what actual musicians would do.

Another repository for folk tunes can be found on *The Session* website[2]. It is an online platform about traditional Irish music which contains more than 46000 transcriptions of monophonic tunes in the ABC format[3].

1 A cleaned version of the Nottingham dataset is available at https://github.com/jukedeck/nottingham-dataset

2 https://thesession.org/

3 The latest data dumps can be found at https://github.com/adactio/TheSession-data and a cleaned version is available at https://github.com/IraKorshunova/folk-rnn/

The lead sheet database

The *Lead Sheet DataBase* (LSDB) [118] is a dataset of jazz and pop songs. It includes nearly all existing jazz song books, which represents more than 15000 music sheets. All lead sheets include a melody together with its chord progression as a sequence of chord symbols. Contrary to the other datasets, they are notated using a markup language similar to the MusicXML format.

Polyphonic datasets

MIDI file collections

Piano-midi.de[4] is a curated collection of classical piano music. It contains selected piano works from 25 composers of the baroque, classical and romantic eras. All files are in the MIDI format. They are not live performance recordings but were entered note after note. Each of the MIDI parameters (pitch, velocity, duration) are controlled so that the rendering of the MIDI sequence sounds like a plausible performance.

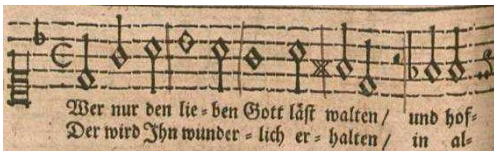
The *MuseData repository*[5] is a collection of instrumental works focusing on music composed in Europe between 1700 and 1825. All files are available in the MIDI format. These files include a variety of genres and instrumentations, from Telemann's solo sonatas for violin to Mozart's symphonies.

Other large MIDI file collections are also available on the Internet. The problem with these files is that their encoding is not always the same and could have been made in order to serve different purposes. For instance, they could be live performance recordings or quantized transcriptions; embellishments can be added or omitted; all voices could be recorded on the same channel or separated between different channels. These large collections can thus be inhomogeneous, not only on the style and instrumentation they feature, but also in the way their MIDI files were devised.

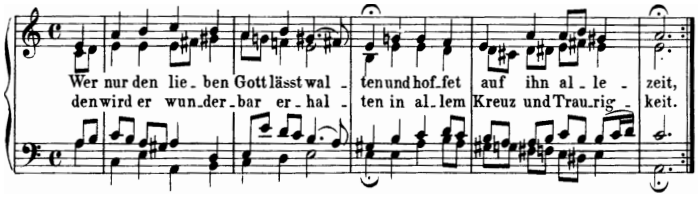
The Chorale Harmonizations by J.S. Bach

The *J.S. Bach Chorales dataset* is the collection of all the chorale harmonizations composed by Johann Sebastian Bach. It is composed of 402 four-part chorales in MusicXML format. This dataset is included in the music21[38] Python package. The corpus of the chorale harmonizations by Johann Sebastian Bach is remarkable by its homogeneity and its size.

4 http://www.piano-midi.de/
5 http://musedata.stanford.edu/



(a) Original text and melody by Georg Neumark (1641),



(b) Four-voice harmonization by Bach: voices are determined by the staff they are written on and the directions of the stems.

Figure 6: Two versions of "Wer nur den lieben Gott läßt walten". The original melody (a) and its reharmonization (b) by Johann Sebastian Bach (BWV 434).

All these short pieces (approximately one minute long) are written for a four-part choir (soprano, alto, tenor and bass) using similar compositional principles: the composer takes a well-known (at that time) melody from a Lutheran hymn and harmonizes it i.e. it composes the three lower parts (alto, tenor and bass) accompanying the soprano (the highest part), see Fig. 6 for an example.

These pieces are usually part of longer vocal pieces like cantatas. Since the aim of the composer is to put a text into music which has to be clearly understandable, all four voices must articulate syllables at the same time. This leads to a particular musical texture called *homophony*. This does not necessarily result in a *homorhythmic* texture, where all voices strictly sing simultaneously, but induces a musical texture in which the variety of rhythm is very restricted. This is also true on a larger scale since all chorales have simple time signatures. Bach chorales mainly focuses on the harmonic and melodic ideas rather than on rhythm. By adding additional voices, the composer "colors" the original melody with new harmonies. This reservoir of harmonic ideas (there often exists several harmonizations for the same melody) makes this dataset a privileged dataset for the study of harmony and musical composition.

From a machine learning perspective, a great asset of this dataset is the presence of fermatas (see Fig. 7) in order to indicate the end of musical phrases. This particular use of the fermata is almost unique to chorale compositions. Knowing where musical phrases begin and end is of a particular interest when building generative models and it is a feature one has to consider.

Figure 7: Fermata symbol.

Composing Bach chorales is a hard problem: it involves combining four-part harmony with characteristic rhythmic patterns and typical melodic movements so as to produce musical phrases which begin, evolve and end (cadences) in a harmonious way. Furthermore, there are some strict musical rules which can sometimes make chorale harmonization looks like a highly combinatorial problem. The language employed by Bach in its chorale harmonizations is rigorous and expressive and subtle. Since it has been studied for centuries, it is easy for an expert to know if a given chorale abides by the style of Bach.

In light of this, the problem of generating chorales in the style of Bach can be evaluated by a human in a rather objective way. For all the aforementioned reasons, most of the experiments presented in this thesis will use this dataset.

3

CHALLENGES IN MUSIC GENERATION

This chapter presents the challenges and issues currently found in the domain of automatic music generation. While the preceding chapter focused on the symbolic musical data itself, we will now observe it from a machine learning point of view with the following question in mind: What are the computational problems raised when building generative models from musical data?

The first problem is the one of finding an appropriate representation for the musical data. We have seen that there exists many musical notation formats, however, these cannot be directly used as the input of our models and need to be converted in an efficient representation. In Sect. 3.1 we present several ways of converting the aforementioned datasets into a representation adapted for learning. We discuss about the common issues and choices faced when crafting a data representation for musical contents. We end this section by proposing our own representation that overcomes some of these issues for some specific musical data.

Once a convenient representation has been chosen, a model, expressive enough to handle the complexity of the musical data, must be devised. We discuss in Sect. 3.2 the many novel and original problems raised by the singularity of the musical data with respect to other types of data.

We then see in Sect. 3.3 that the lack of objective metrics in order to assess the quality of the generated pieces can be a major issue when devising generative models. This chapter concludes on the following question: What these generative models of music can be used for? We will see in Sect. 3.4 that having a clear objective condition how we conceive these generative models.

BUILDING REPRESENTATIONS

As seen in Chap. 2, an intermediate representation is thus needed. Unfortunately, as for musical formats, no representation is universally acknowledged as being the most effective nor can handle the diversity and specificities of all types of music. We now describe encodings for the different parameters of the music, how they are related to each other and discuss their advantages and drawbacks.

Notes

Choosing the right encoding for the notes is important. The most-used encoding is the MIDI pitch encoding (Sect. 2.1.4). That is: each note on a piano is given an integer value and each one is seen as a distinct category. Nearly all existing approaches are built using this encoding. However, we claim that it is not the right choice when dealing with music sheets.

Indeed, when learning from music sheets so as to generate music sheets, we need the output music sheet to be well-written so that it can be played, for instance, by a musician. This is impossible using the MIDI pitch encoding. The reason for this is that there is a loss of information when reducing the *full name* of the notes (like C♯4, E♯5 or B♭4 for instance) to their MIDI pitches because of the *enharmonic* notes. From a musical point of view, loosing this information is problematic since most of the music we know makes this distinction (see Sect. 2.1.1). Generated music sheets that use this encoding would necessarily have *spelling* problems and would require additional treatment before being considered as "correct". A more problematic fact is that this loss of information on training data can in fact hinder the model performances since we decide, with no clear reason, to totally ignore the implicit presence of music theory in the data. For instance, this difference in Bach chorales is unambiguous and it is thus natural to consider the full name of the notes. From a machine learning point of view, two enharmonic notes would appear in totally different contexts: considering them as being identical would force the model to learn to distinguish between both, which may be hard considering the scarcity of the symbolic music datasets. I believe that this encoding is essential in order to allow generative models to generate notes with the correct spelling, which is of the utmost importance when we focus on the music sheet rather than on its audio rendering.

However, as seen in Sect. 2.3, the biggest datasets are in MIDI format, which do not contain this basic information. We thus advocate for the need of more symbolic datasets using the open MusicXML format.

Rhythm

Symbolic music represents the organization of sounds through time. As seen in Sect. 2.2.1, even simple melodies have a complex and hierarchical organization and a convenient encoding of rhythm is essential.

There are mainly two ways to treat rhythm:

- using a event-based encoding,
- using a fixed time grid.

Event-based encoding

This encoding consists in considering that music is composed of notes with a prescribed starting and ending time or with a starting time and prescribed duration. Examples of formats following this scheme are:

- the MIDI format (Sect. 2.1.4),
- the ABC notation (Sect. 2.1.3).

Note that it is possible to slightly modify the MIDI format so that, instead of having *note on* and *note off* events, we have only information about which note is played together with its starting time and its duration. When there is only one voice, the information about the starting time can be omitted since notes succeeds to one another and we end up with an encoding very similar to the ABC notation.

In both cases, a musical event can be seen as a couple of a note together with a timing information.

If this is apparently the most natural way of encoding music (since it is more or less a straightforward adaptation of our commonly-used musical notation (Sect. 2.1.1), it is not necessarily appropriate when used in generative models. As an example, in a correct MIDI file, a *note on* MIDI event will always be followed by a *note off* event. For a generative model, learning such a structure adds unnecessary complications and can lead to the generation of ill-defined sequences (see [71] for a discussion on the advantages and drawbacks of using the MIDI encoding for generation tasks). The great advantage of this encoding is that it is able, in theory, to encode *any* rhythm: from crotchets to more complicated rhythms such as irrational rhythms. However, since in symbolic music generation the duration of the rhythms we have are only rational (compared to an encoding of a true performance), we can think about more efficient ways of encoding the duration (rather than trying to predict real numbers). This implies for instance to discretize time using the smallest possible subdivision. This is not the only way of representing rhythm (see for instance [108] where duration is represented using a five dimensional space).

There are also other difficulties provoked by using this encoding. One, for instance, is that a generative model would have to predict couples composed of a note and its duration. How to consider this? Is this a Cartesian product between all the possible notes and all the possible durations or simply consider them as independent from each other? In a word, this adds modeling questions since we need to model a joint distribution over couples. This is important since it directly affects how a generative model can train. Another difficulty stems from the fact that if a "wrong" duration is produced at generation time, this error will affect all the following notes since they can for instance all become "out of beat". This is why generative models must also learn metrical information such as measure bars to avoid

these drawbacks (see for instance the method proposed in [145] and discussed in Sect. 4.1.1.1).

Fixed time grid encoding

Another way of thinking about music is to consider that time unfolds regardless of the presence/absence of notes and that for each given time t, we can list all the notes being played. The major difference with the preceding approach is that there is no more time/duration coupling. Concretely, time is often discretized and the set of notes being played is represented using a binary vector. This leads to a grid-like or piano roll encoding of the music (see Fig.8).

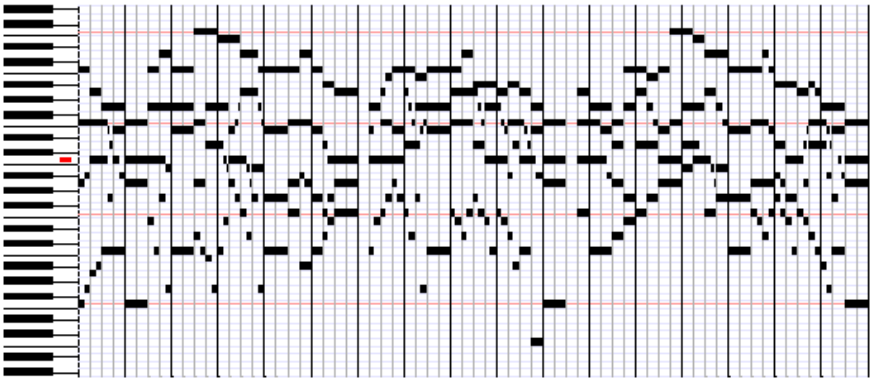


Figure 8: Example of a piano roll where notes are encoded using MIDI pitches.

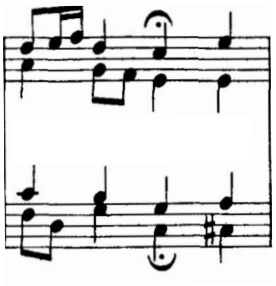
Discretization can cause some issues. For instance, how to choose the minimal subdivision? This can be problematic if the corpus contains tuples of different sizes or really fast rhythms. In such a case, the encoding is mostly inefficient and it is hard to learn from musical data encoded using this scheme: approximations and trade-offs must be made.

The usual piano roll has two major drawbacks:

- there is no distinction between voices,
- there is no distinction between a held note and a repeated note.

Indeed, the piece in Fig. 8 seems to be written for four monophonic parts. However, this information is lost and it is only possible to guess. In such cases, a common solution is to put voices in different piano roll tracks. This is more consistent with the way music is written: the distinction between instruments or between hands (for keyboard pieces) is always clear on the music sheets.

The second issue, which is that we are unable to distinguish between a held note and a repeated note, is more problematic since it profoundly alters the music. In order to circumvent this problem, we can add, following [74, 150], an additional piano roll of Boolean



(a)

```
D5,__,E5,F5,D5,__,__,__,C5,__,__,__,E5
A4,__,__,__,G4,__,F4,__,E4,__,__,__,E4
C4,__,__,__,B3,__,__,__,G3,__,__,__,A3
F3,__,D3,__,G3,__,__,__,C2,__,__,__,C#2
```

(b)

Figure 9: Extract from a Bach chorale and its representation as four voice lists. The hold symbol is displayed as "__" and considered as a note.

values which would encode if a note is articulated or just held. As a consequence, the generative model must generate two elements: a note together with its articulation.

If most of the encodings used in music generation are related to the ones described above, many add other extra information depending on the dataset they train on.

Melodico-rhythmic encoding

Both approaches shown in Sect. 3.1.2 had in common the fact that they had to generate couples of values (note and duration or note articulation couples). We think that this is a disadvantage since it forces to model particular joint distributions over pairs.

We now propose a unified encoding for monophonic sequences which allows to treat melody and rhythm on an equal footing. In the following, time is discretized and we have a fixed time grid.

This encoding relies on a simple trick. This is done by adding a *hold symbol* "__" to the list of possible notes. This symbol codes whether or not the preceding note is held. Since the sequences we consider are monophonic (only one sound at a given time), this representation is unambiguous and compact.

Furthermore, it is not only limited to monophonic music but can be used in every music piece composed for distinct monophonic voices. Its melodico-rhythmic encoding is then obtained by taking the melodico-rhythmic encoding on each monophonic voice (see Fig. 9 for an example).

The requirement that all voices be monophonic is not so restrictive, since this embraces music from chorale music to orchestral works (if we throw away double stops).

This encoding will be used in all the experiments presented in this thesis. In these experiments, the generative models I propose benefit from the advantages offered by the melodico-rhythmic encoding

in totally different ways. We will discuss in detail this aspect in the corresponding sections.

COMPLEXITY OF THE MUSICAL DATA

As seen in Sect. 2.2.2, musical data is complex and has greatly evolved throughout ages. It is not however the only source of variability in musical data: even if we choose only works by a single composer or belonging to the same musical period, the variability between different musical pieces is important. There are variations in genres, tempi, time signatures, key signatures, instrumentation, length, compositional processes, etc. The Cello Suites by J.S. Bach are extremely different from its great preludes and fugues for organ. And even within these more homogeneous subcategories, strong and crucial differences still exist: compare for instance a prelude and a sarabande from the Cello Suites. Only this simple observation raises important questions:

- Is it possible to learn from data which is so heterogeneous?
- If this is possible, what would a trained model be able to generate?
- If we restrict ourselves to homogeneous (but really small) datasets, is it possible to generalize well?

I believe that this trade-off between the size of the datasets and their coherence is one of the major issue when building deep generative models. If the dataset is very heterogeneous, a good generative model should be able to distinguish the different subcategories and manage to generalize well. On the contrary, if there are only slight differences between subcategories, it is important to know if the "averaged model" can produce musically-interesting results.

There are nonetheless techniques that reduce some factors of variability (by transposing all pieces in the same key [17, 93]) or, on the contrary, artificially augment the size of the datasets (by transposing the dataset in all keys [87]). However, for the majority of the other factors of variability, not much can be done.

Another issue is that each musical piece is self-contained and can have a lot of structure (long and short-term structure). For some musical works like sonatas or fugues, the global structure can indeed be at the crux of the composition. To generate a convincing sonata, not only the language must be learned from data but also the ability to create long, balanced and coherent pieces.

These difficulties create many challenging problems like:

- How to devise data-efficient and expressive models?
- Can/should we include musical knowledge to improve the generation capabilities?

- Which aspect of the data can we try to reproduce (patterns, variations, harmonies, structure) and how to achieve this goal?
- Can a generative model generate in a single shot a fully coherent, long and interesting musical piece?

The careful choice of the dataset is therefore essential. Some, for instance, are "easier" to learn from than others. Depending on the music at stake, the musical interest can rely more on some parameters than on others. For instance, structure, rhythm and patterns are predominant notions in folk music 2.3.1.1; harmonies, on the contrary, are to be simple, with clear harmonic functions. Another example is the corpus of the chorales by J.S. Bach 2.3.4. In this case, harmony plays a central role while long-term structure is mostly dictated by the melody to harmonize.

Understanding the musical distinctions between datasets is thus the very first step when building generative models. Furthermore, some styles may be more permissive than others: in some styles, inaccuracies in a generated piece may only slightly affect the musical quality of the whole piece or go unnoticed while in some others, slight mistakes can ruin the whole piece.

EVALUATION OF GENERATIVE MODELS

Generative models are often hard to evaluate using objective metrics. This is all the more true when considering deep generative models for symbolic music. If there exists many works on music theory [80, 126], there rarely exists musical rules that apply to all musical pieces.

It is thus impossible to have quantitative measures of accuracy or to compare models. There is no MNIST-like task for music generation and better log-likelihood results do not imply better musical generations [147]. This causes many hardships since designing and improving a model can only be made in a blindfolded fashion. We cannot for instance optimize models hyperparameters on a both objective and musical basis.

One way to evaluate or compare models is through perceptual listening tests. With the recent improvements of the quality of the generated pieces, it is now possible to devise Turing-like tests, where the objective for the participants is to determine whether or not the musical pieces are computer-generated [62, 93]. If this is a good practice, the results are very noisy and cannot be used as a ground base.

Another way is to simply ask experts to rate the musical quality of the generated pieces. Through music analysis, it is possible to analyze the advantages and drawbacks of the proposed methods. However, a major difference with other types of data often considered (like text or images) is that a real expertise is needed for this task.

A feature often put aside when evaluating results about generative models is their ability to generate new content. Even if it is rarely considered, it is an absolute necessity that a generative model does not plagiarize its training dataset.

GENERATIVE MODELS FOR MUSIC, WHAT FOR?

The preceding sections seemed to be rather pessimistic about the possible achievements in music generation. In our view, we believe that thinking about the human-machine interactions for improving composition rather than thinking about generating perfect musical pieces from scratch can help overcome some of these issues and provide new paths of research.

Bearing this objective in mind radically changes the way to think about automatic symbolic music generation. The generative models are now only devised to be used for specific tasks and part of a greater interactive system. Such a system must be intuitive and powerful, but need also to let the stylistic choices up to the composer. Doing so breaks into smaller parts an extremely complex problem. The aim is now to make generative models able to propose interesting ideas. From a computational point of view, we no longer have to generate from scratch, but need to improve over given solutions. Sometimes, only one suggestion of a good chord is sufficient to transform a whole song.

This will be the guide line of this thesis. For instance, the DeepBach model (Chap. 6) stems from the following interrogation: What if I am satisfied with the beginning and end of a generated chorale but would like to change the middle part? Most of the existing systems do not allow such a feature. Now suppose that you have an existing melody, but you would like to ornate or vary it. Can we devise a system able to do this? What kind of control can we have on this regeneration process? These ideas are developed in Part III.

The evaluation of interactive deep generative models becomes simpler: an interactive deep generative model is good if it helps to generate with little effort music that we can listen to and appreciate. The compositional process resulting from the use of such a system must be intuitive and let the composer play an active part in the composition. In order to achieve this goal, the generative model must be flexible enough to be capable of coping with many of the user's queries so that a balanced and fruitful dialogue between the composer and the machine can take place.

4

DEEP LEARNING MODELS FOR SYMBOLIC MUSIC GENERATION

This chapter is an exposition of the recent approaches in music generation. We do not seek to be fully exhaustive but prefer to give a glance at the different methods, of the problems they solve and on the issues they suffer from. This presentation is mainly focused on the recent data-driven deep learning generation methods.

For a more comprehensive survey, we refer the reader to [48]. The authors of this paper review all methods in algorithmic composition up to 2013. However, since then, the generalized use of neural network-based techniques in this domain has (almost) outpaced every other approaches and no unified attempt at analyzing the use of deep learning methods for generating music has been published yet. This chapter can thus be considered as a first step in synthesizing and analyzing the novel ideas from the machine learning community to solve original music-specific problems. For a more complete classification of the deep learning generative techniques for music, we refer to the recently published survey [19].

The objective here is to give a brief overview of the existing deep generative techniques applied to symbolic music. We will try to discuss how similar ideas are shared among apparently different approaches. We will then see in the rest of this thesis how thinking about adding interactivity to these models help to design well-motivated and useful generative models.

SEQUENTIAL MODELS

Music unfolds through time and seems thus inherently sequential. Recurrent Neural Networks (RNNs) [58] appear as a natural solution for a probabilistic modeling of musical data. This section presents several recently-introduced generative models. Even if they differ from each other on the dataset they train on, the representation of the data they use or on their overall architecture, they all share the same characteristic: they generate music from left to right. However, this does not match with real compositional processes. A composer almost never writes music from left to right but would rather start by writing themes and harmonizing them, construct a harmonic skeleton for the whole piece or maybe try different accompaniments.

Models on monophonic datasets

Folk-RNN model

The folk-RNN model is a model introduced in [145] whose aim is to generate Celtic melodies (Sect. 2.3.1.1). It is based on the ABC notation (Sect. 2.1.3) in the sense that it learns and generates ABC notations. Since folk songs are encoded using text, it seems natural to apply generative models designed for text directly on the folk songs in ABC format. In fact, the authors use an appropriate tokenization of the dataset in order to discard unnecessary redundancy coming from the encoding itself: for instance, the `:|` symbol, the `d'` note or the `/2` duration are considered as one single token. This has the effect to improve the quality of the generations compared to a plain character-based encoding.

For a given sequence of tokens $s = (s_1, \dots, s_l)$ of length l, the folk-RNN model consists in a discrete conditional distribution

$$p(s_t|s_{<t}) \tag{1}$$

over the set of all possible tokens, where we note by $s_{<i}$ the sequence $(s_1, \dots, s_{i-1})$. This amounts to factorizing the probability distribution of a whole sequence as

$$p(s = (s_1, \dots, s_l)) = \prod_{t=1}^{l} p(s_t|s_{<t}), \tag{2}$$

with $s_{<1} = \emptyset$. The authors use a stack of three Long Short-Term Memory (LSTM) [58, 69] networks for an effective implementation of the conditional distribution of Eq. 1. Training this recurrent network is then done in an iterative way using the backpropagation through time (BPTT) algorithm [109, 158].

Generating a sequence with such a model is easy and fast. Given a seed $s = (s_1, \dots, s_{t-1})$ of size $t-1$, we generate the next token s_t by sampling from the categorical distribution of Eq. 1 and we can generate a sequence of any size by repeating the preceding step over and over.

It is interesting to note that in this approach the bar lines and the repeat bar lines are given explicitly and are to be predicted as well. This can cause some issues since there is no guarantee that the output sequence of tokens s would represent a valid song in ABC format. There could be too many notes in one bar for example, but according to the authors, this rarely occurs. This would tend to show that such an architecture is able to learn to count [54].

The results are very convincing in the sense that the generated melodies seem to be typical Irish tunes (see Fig. 10). One may see and listen to the results on the following site [143]. The authors also evaluate the novelty of the generated sequences by checking if there

exists subsequences that are directly copied from the training dataset. The conclusion is that even if small subsequences can be found verbatim in the training dataset, these subsequences are recombined in such a way so that it produces brand new melodies. Often, these subsequences are in fact characteristic elements or common patterns in folk music. Such a behavior is thus desirable since it allows the Celtic style to be recognizable while creating new musical material. Besides, the generated pieces have been played by traditional Irish music players. This is, in a way, an empirical validation of the quality of the generated pieces.

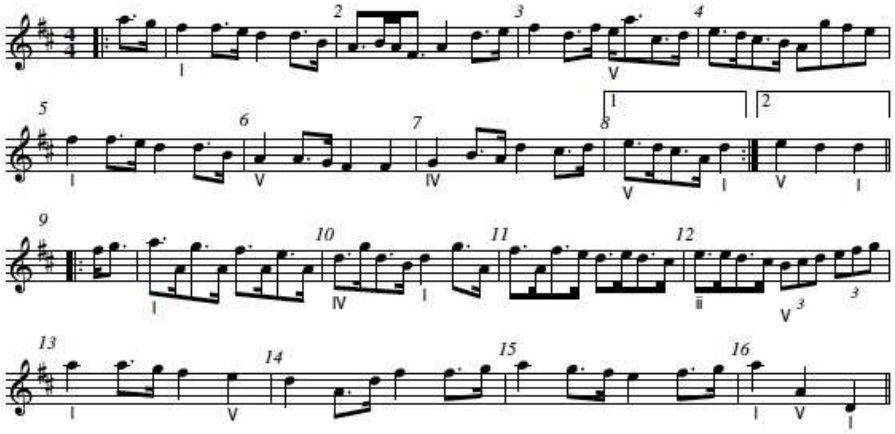


Figure 10: Score of "The Mal's Copporim" automatically generated. Reproduced from [145]. Degree annotations as analyzed by the authors are added on the bottom of each staff.

However, one has very little control over the outputs. The only way a user can steer this model is by providing the beginning of a musical sequence that the model would have to continue. Since key signature and time signature information are always given in the preamble of each sequence, this can induce some interesting and non trivial ways to condition generation of the output sequences. Nonetheless, interactivity is very limited: after a sequence has been generated, a user can only keep it or discard it.

Sequence Tutor

We have just seen that training a deep RNN on an appropriate dataset can produce seemingly good results. However, what happens if this does not work as expected? If, for instance, the generated sequences do not fulfill some precise musical rules, can we devise an algorithm to improve our generative model and enforce these rules? This problem is addressed in [73] where the authors propose to introduce Reinforcement Learning (RL) techniques in order to obtain a model able to take into account musical principles while learning from a dataset of melodies. The dataset that they consider contains the melodies from 30000 MIDI songs (Sect. 2.3.3).

Their architecture, called Sequence Tutor, cast the iterative left-to-right generation of tokens as a reinforcement learning problem. In this setting, we have an agent that interacts with an environment e following a policy π. At time t, performing action a_t in the environment e_t grants a reward $r(e_t, a_t)$, where r is a function to define. Given a sequence of actions (a_t) and environment states (e_t), the quantity of interest is the return

$$R := \sum_{t=1}^{\infty} \gamma^t r(a_t|e_t),$$

defined as the discounted sum of all rewards, with discount factor $\gamma < 1$. The aim is then to learn a policy $\pi(a_t|e_t)$ which maximizes the expected return $E_\pi[R]$, where the a_t are sampled using $\pi(a_t|e_t)$.

In the context of sequence generation, performing action a_t at time t consists in generating the t^{th} sequence element s_t. The state of the environment e_t up to time t is the sequence of tokens $s_{<t} = (s_1, \ldots, s_{t-1})$ generated so far. The interesting problem which arises when reformulating the left-to-right generation consists in finding a good reward function $r(s_t|s_{<t})$. Ideally, the authors want a reward function that would take into account both the training dataset (style) and some prescribed musical rules. The difficulty here is that musical rules are inherently discrete values and involve a whole sequence of notes to be evaluated rather than only single notes. Examples of such rules are: sequences must start with the tonic, sequence must go down after a leap up, there should be only one highest and one lowest note, etc. These rules are used to build a handcrafted reward function r_T based on music theory.

An easy way to take into account the training dataset is by maximizing the log-likelihood of a RNN as it is done in Sect. 4.1.1.1. This gives rise to a trained probabilistic model $p(s_t|s_{<t})$ as in Eq. 1 that the authors name Note RNN. Since the log-likelihood of a sequence is the sum of the transition log-likelihoods

$$\log\left(p(s = (s_1, \ldots, s_l))\right) := \sum_{t=1}^{l} \log p(s_t|s_{<t}), \tag{3}$$

taking $r_L(s_t|s_{<t}) := \log p(s_t|s_{<t})$ for a reward can then help obtaining policies that generate sequences of high probability according to the Note RNN model.

The authors then naturally consider a blend of the two preceding reward functions, namely

$$r(s_t|s_{<t}) := r_L(s_t|s_{<t}) + r_T(s_t|s_{<t})/c, \tag{4}$$

between the Likelihood-based reward and the music Theory reward. There is an additional trade-off constant c which balances between the fidelity to the style and the respect of the musical rules. The idea

is to slightly deviate from the original "style" policy while enforcing some constraints. Training is performed using a generalization of the usual Deep Q Networks (DQN) [106] called ψ-learning [128]. Roughly, this allows the learned policy π to be non-deterministic (as it would be the case with the policy obtain by a DQN) by considering an entropy-regularized objective during training. A stochastic generation process is indeed an essential requirement in music generation. Note that the Note RNN model is considered to be fixed during the learning of the policy. The RL-Tuner architecture is illustrated at Figure 11.

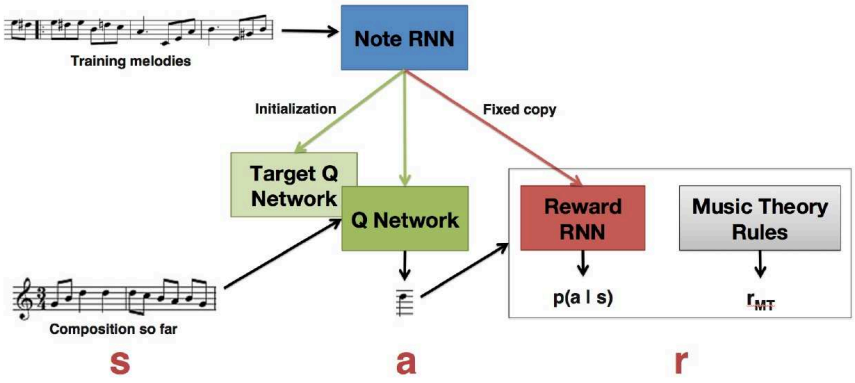


Figure 11: RL-Tuner architecture. From [73].

The authors report positive results on how this method is able to enforce the musical rules and help improve their original Note RNN generations. The possibility to tweak an existing model so that it can enforce complex constraints sounds appealing, but also raises other questions. Why the original Note RNN model cannot learn these rules from data? Furthermore, the choice of the musical rules seems quite arbitrary and the musical aim is thus unclear. In our opinion, this stems from the choice of the corpus which might be too heterogeneous to yield an interesting Note RNN model. Indeed, contrary to the model proposed in [145] or the ones proposed in this thesis, the training data possesses no definite style and thus no precise musical evaluation of the output is possible. In such a setting, the Note RNN can be enhanced in order to consider some musical principles, but in the end, we do not think that a clear musical style can still emerge from this approach.

The "control" that we have here is only over the choice of the musical rules, and this must be done before the training of the model. An interaction between a user and the system at generation time cannot be devised.

Improving left to right generation

In all these sequential approaches, some refinements in the sampling procedure can be made. This often consists in tweaking a learned

RNN model as in Eq. 1 so that some unwanted behaviors disappear. Generating a "wrong" note can indeed have implications on the remaining of the generated sequence.

A simple trick in order to avoid the generation of "wrong" notes (defined to be notes generated with low probability) is to add a threshold parameter in Eq. 1. Following [159], this equation becomes:

$$p_{\text{threshold}}(s_t|s_{<t}) := \begin{cases} 0 & \text{if } p(s_t|s_{<t})/m < t, \\ p(s_t|s_{<t})/Z & \text{otherwise,} \end{cases} \tag{5}$$

where $m = \max_{s_t} p(s_t|s_{<t})$ is the maximum transition probability, t the threshold parameter and Z a normalization constant.

Similar to this approach is the introduction of an *inverse temperature* parameter $\beta \geqslant 0$. Concretely, it consists in using a learned model as in Eq. 1 to build a new model

$$p_\beta(s_t|s_{<t}) := \frac{\exp(\beta \log(p(s_t|s_{<t})))}{\sum_{s'_t} \exp(\beta \log(p(s'_t|s_{<t})))}. \tag{6}$$

For $\beta = 1$, $p_\beta = p$. However, in the low temperature regime ($\beta > 1$), some probability mass is transferred from transitions with low probability to transitions with high probability. This approach has the advantage to smoothly interpolate between the original conditional probability distribution and the argmax deterministic distribution

$$p_{\text{argmax}}(s_t|s_{<t}) = \text{argmax}_{s'_t} p(s'_t|s_{<t}). \tag{7}$$

On the contrary, using p_β with $0 \leqslant \beta < 1$ smoothly interpolates between the original conditional probability distribution and a uniform distribution ($\beta = 0$).

These techniques can sometimes be used to slightly improve the musical results of the generated sequences by producing more "obvious" choices (in the case $\beta > 1$).

In some approaches, the aim is not to sample sequences from Eq. 2 with the correct probabilities but rather to generate sequences with high probability. In this case, the generation of a sequence consists in finding a path of large weight in the graph whose nodes are indexed by the subsequences $(s_1, \dots, s_t)$ for varying t and whose edges between $(s_1, \dots, s_{t-1})$ and $(s_1, \dots, s_t)$ are weighted by $\log p(s_t|s_{<t})$. Such an approach is however highly combinatorial and some heuristics have to be chosen. The beam search heuristic algorithm is often used in such a case. Examples of its usage as well as practical considerations about beam search can be found in [20].

Polyphonic models

We have seen that deep generative models for monophonic music tend to use a RNN in order to model the probability distribution of

the next token given all previously-seen tokens. Generation is then done iteratively by simply sampling tokens one at a time. When considering polyphonic music, this probability distribution is no longer a simple histogram over all the next possible notes but a more complex distribution. In some of the dataset considered, an arbitrary number of notes can be played at the same time. The approaches we present here consider different ways of modeling these complex distributions over chords composed of an arbitrary number of notes.

RNN-RBM model

The RNN-RBM model introduced in [17] proposes the following approach in order to handle music encoded using the piano roll representation. The datasets they consider are, among others, the Piano-midi.de (described in Sect. 2.3.3) and the J.S. Bach chorales dataset (see Sect. 2.3.4). For each time t, there can be any number of notes and the idea is to have a parametrized distribution which allows to sample from this complex distribution. The Restricted Boltzmann machine (RBM) [68] allows to learn a probability distribution in an unsupervised manner and to efficiently sample from it. We note $v^{(t)} = (v_1^{(t)}, \dots, v_{128}^{(t)})$ the notes being played at time t, where each $v_i^{(t)}$ is a Boolean value indicating that note with MIDI pitch i is being played. In order to model a complex distribution over $v^{(t)}$, the idea behind the RBM is to introduce a vector of Boolean hidden units $h^{(t)}$ on which depends the visible units $v^{(t)}$. Their interaction is modeled by the following joint probability distribution

$$P(v^{(t)}, h^{(t)}) = \frac{1}{Z} e^{-E(v^{(t)}, h^{(t)})}, \tag{8}$$

where Z is a normalizing constant and where the energy function $E(v^{(t)}, h^{(t)})$ is given by

$$E(v^{(t)}, h^{(t)}) = -(b_v^{(t)})^T v^{(t)} - (b_h^{(t)})^T h^{(t)} - (v^{(t)})^T W h^{(t)}, \tag{9}$$

with visible and hidden biases $b_v^{(t)}$, $b_h^{(t)}$ being real vectors, W the interaction matrix and where ^{T} denotes the transposition . Note that there is no interaction between hidden units nor between visible units contrary to the more general Boltzmann machine: there is no

$$(h^{(t)})^T W_h h^{(t)}$$

term nor

$$(v^{(t)})^T W_v v^{(t)}$$

term in the energy functional.

The probability of the vector $v^{(t)}$ is then obtained by marginalizing over all hidden units which gives us

$$P(v^{(t)}) = \frac{1}{Z'} \sum_{h^{(t)}} e^{-E(v^{(t)}, h^{(t)})}. \tag{10}$$

This is generally computationally intractable but efficient sampling and inference algorithms exist (e.g., the Contrastive Divergence algorithm [148]).

This RBM models the probability distribution of a time slice at time t. It remains to model the sequential aspect of the piano roll, namely: how these distributions evolve through time?

The idea of the authors in [17] is to consider that the dependency on time is only accounted for by the vectors $b_v^{(t)}$, $b_h^{(t)}$. On the contrary, the W matrix is shared across all time steps t. To this end, a RNN is introduced. This RNN models the time dependency of meta-variables $u^{(t)}$ which influence variables $b_v^{(t)}$, $b_h^{(t)}$ through the introduction of matrices W_{uv} and W_{uh}. The authors describe efficient training and sampling schemes for this architecture, a sketch of which is shown in Fig. 12.

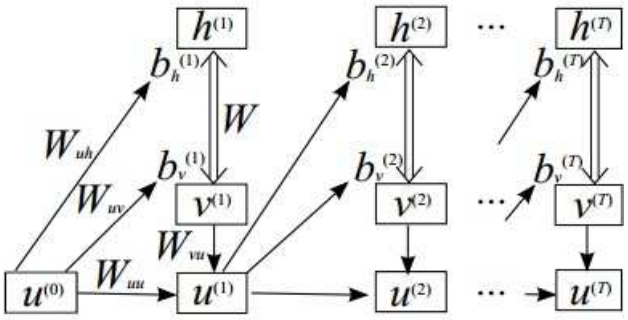


Figure 12: RNN-RBM architecture reproduced from [17]. The RBM is depicted using double arrows.

The interesting aspect in their approach is that they model music by considering it as a succession (*horizontal view*) of time slices (*vertical view*) where each aspect is modeled by a different neural network architecture. This model has many generalizations, depending on whether we change the time slice probability density estimators or the way to implement the recurrent temporal part. For instance, a model where the RBM part is replaced by a Neural Autoregressive Density Estimator (NADE) [85] is also evaluated in [17]. Replacing the RBM part by a Deep Belief Network (DBN) [67] is studied in [57]. Changing the implementation of the RNN has also been tested. We can cite for instance the modeling in [95], which implements the RNN part using LSTM units, or the one in [30] which implements it using Gated Recurrent Units (GRU) (an LSTM variant).

If this model is very appealing because it can theoretically model any type of music in a piano roll representation, the musical evaluation of the generated sequences reveal some issues. On the MIDI databases, no clear style really emerges and so we cannot have any idea of what the model actually learned. On the more homogeneous corpus of the J.S. Bach chorales, the results are more convincing, but

could not been perceived as original Bach chorales. In our opinion, the reason is that this approach may be too general for such a specific problem. Many additional information could be taken into account (see Sect. 2.3.4), like the fixed number of voices, the difference between voices, the true spelling of the notes or the fermatas. The model has thus too many things to learn and too few data.

Biaxial RNN

We now mention an approach which proposes another way of modeling the *vertical view* of the piano roll representation. The idea behind the Biaxial RNN [74] is to use the same model for both the vertical and horizontal views of the music. This results in a more symmetrical model contrary to the previous approach which clearly differentiated between both. This architecture is presented in Fig. 13.

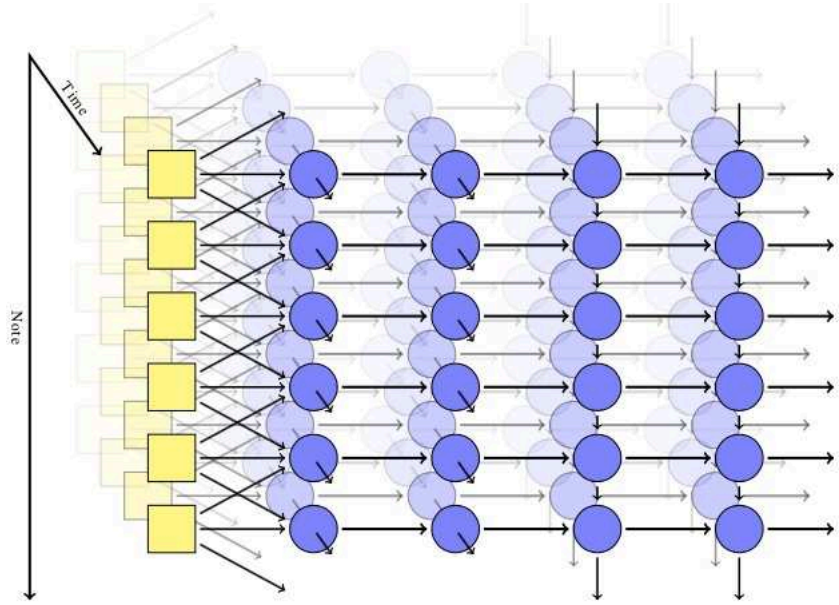


Figure 13: Biaxial RNN architecture. Each column has its own LSTM unit represented as a blue circle. The 2-layer Time-LSTM is represented by the first two columns while the 2-layer Note-LSTM is represented by the two last columns. Reproduced from [74].

It consists in two intertwined (stacked) LSTM networks: one is in charge of modeling the time dependence of a given note between different time steps while the other one is in charge of modeling the dependence between notes played during the same time step. A LSTM network is invariant with respect to the dimension it is consecutively applied on: the same unit is used iteratively on each input element and the time dependence is contained exclusively in the hidden state. Therefore, applying a LSTM on the vector $v^{(t)}$ (keeping the notation introduced in the previous section) is supposed to be invariant with respect to transpositions. Combining both would lead to

a model capable of learning and generating transposition-invariant patterns. The advantage of this modeling is that the inference procedure is no longer approximate in this case. The author trains on the piano-midi.de dataset and reports good generation results. Generated pieces are coherent, but lack of overall structure. It is interesting to note that additional information has been added to the piano roll representation, so that

- the beat is taken into account by adding a binary representation of the position within a bar,
- the ordering between notes is considered (by providing the MIDI pitch value, the system is able to know if a note is high or low),
- the distinction between a held note and a repeated note is made.

BachBot

We finally present a specialized approach on the J.S. Bach chorales dataset called BachBot [93]. The idea here is to use only one RNN for both the horizontal and vertical view of the data. Using only one RNN for apparently inherently multidimensional data has indeed proved to be successful on image generation tasks [113]. The idea consists in choosing a particular ordering of the notes so that one can unfold a chorale composed of multiple voices into a sequence. Contrary to the previously cited approaches in polyphonic music generation, this work does not rely on a piano roll encoding, but considers each voice separately. The unfolding is done by going from top to bottom and from left to right. More precisely, for a given time step, the model first considers the note played by the soprano part, and then (in the following order) notes played by the alto, the tenor and the bass parts. Similarly to the introduction of an "end of measure symbol" described in the folk-RNN model (Sect. 4.1.1.1), a special character ||| is added in order to indicate the end of a time slice. This gives landmarks to the model: this information about the voice being generated helps the model to better differentiate between the different voices. The specific use of the fermata in Bach chorales (see Sect. 2.3.4) is also taken into account. In this approach, all chorales are transposed into the same key of C major/A minor.

An example of the resulting encoding is shown in Fig. 14.

The BachBot model is then very similar to the folk-RNN model, but working on a different encoding and a different dataset. The results are extremely convincing so that an online Turing test has been conducted in order to see if the participants could distinguish these generations from original Bach generations.

As for the folk-RNN model and all the approaches presented so far, this method cannot however be used in an interactive fashion.

```
(74, True)
(65, False)
(59, True)
(55, True)
|||
(.)
(72, False)
(64, False)
(55, False)
(48, False)
|||
(.)
(72, True)
(64, True)
(55, True)
(48, True)
|||
```

Figure 14: BachBot chorale encoding of part of the extract displayed in Fig. 9. Adapted from [93].

The authors have nonetheless devised a simple modification of their algorithm so that it can be used to reharmonize melodies: at generation time, they do not sample the soprano part but instead use the user-given melody. This approach gives good results even if it is not satisfactory from a theoretical point of view since the model cannot "anticipate" what the future soprano notes will be.

AUTOENCODER-BASED APPROACHES

There are nonetheless approaches which try to generate sequences all at once rather than note by note. The methods that we describe in this section try to *encode* a whole sequence s into one point (or latent space representation z) in a space of small dimensionality. This latent space can be considered as the space of all (valid) sequences. A mapping (*decoder*) from the latent space to the space of sequences is then introduced in order to generate sequences given a latent space variable z. The encoding and decoding functions are jointly learned: the aim is to perfectly reconstruct the sequence which has been encoded. Since the latent space is of smaller dimensionality than the space of all sequences, data-relevant codes and efficient decoding functions must be found. This is the original idea motivating the autoencoder architecture and its refinements [13, 153].

When transposed into the context of sequence generation, the encoding and decoding functions are often implemented using RNNs which are convenient when dealing with sequential data (see Fig. 15). Sampling from an autoencoder is easily implemented: it suffices to

draw a random latent variable z and to decode it in order to get a meaningful sequence. However, this sampling scheme is not satisfactory since we have no guarantee that we sample sequences with the correct probabilities.

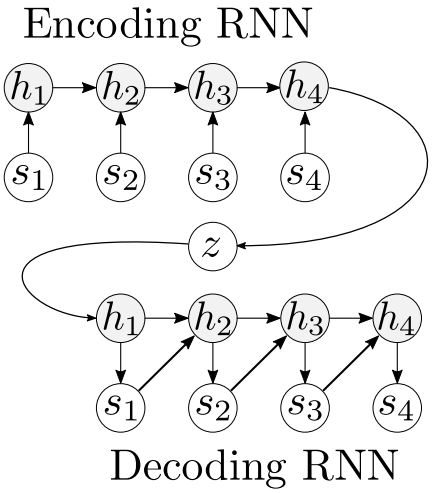


Figure 15: Autoencoder architecture for sequence generation. Adapted from [47]. The input sequence $s = (s_1, \ldots, s_4)$ is mapped to the latent variable z using the encoding RNN. It is then decoded using the decoding RNN. The variables $h_1, \ldots, h_4$ are the RNNs hidden states.

Variational Autoencoder for MIDI generation

The Variational AutoEncoder (VAE) framework [78] adds a probabilistic point of view to the autoencoder architecture. Basically, it introduces the possibility to learn the data distribution and to sample from it. This is done by imposing a prior distribution $p(z)$, often a normal distribution, over the latent space. The decoding function is written as the conditional probability distribution $p(s|z)$ so that the probability of a sequence s is

$$p(s) = \int p(s|z)p(z)dz. \tag{11}$$

See Sect. 9.2.1 for a more precise presentation of the variational autoencoder framework and its training objective.

The autoencoders perform dimensionality reduction and can be seen as a non-linear generalization of Principal Component Analysis (PCA). The advantage of the variational autoencoder over the autoencoder is that the latent space learned by a variational autoencoder has a less "abrupt" behavior. In the settings of the variational autoencoders, the aim is not only to be able to achieve perfect reconstruction but also to be able to sample from the data distribution. This forces

the model to learn a smoother latent space representation and prevents some overfitting issues. Neighboring points in the latent space are more prone to look like neighboring sequences in the space of musical sequences. This is further discussed in Sect. 9.1.

This model is first used in [47] in the context music generation. They apply it on 8 MIDI files of 80s and 90s video game songs. The dataset is composed of non-overlapping subsequences of length 50 from these songs. As a consequence, this model is only able to produce sequences of length 50. Generating music of fixed length appears to be great limitation of this model, however, the interest of this architecture relies in the embedding that is learned. The possibility to map a continuous latent space into the discrete space of correct sequences grants the latter with a geometrical interpretation and suggests novel applications. For example, since neighboring points in the latent space are mapped to sequences which are "close" to each other, we can use this model to generate slight variations of a given musical chunk by exploring its neighborhood (once encoded into the latent space). Also, interpolating between two discrete sequences in a "continuous" way is made possible.

These original techniques provide new insights on the input data and can be used for creative purposes. For example, this has been applied on text in [18] where the authors generate sentences by interpolating between two sentences. Figure 16 shows how the original musical sequences from the dataset are mapped into the latent space using the learned encoding function.

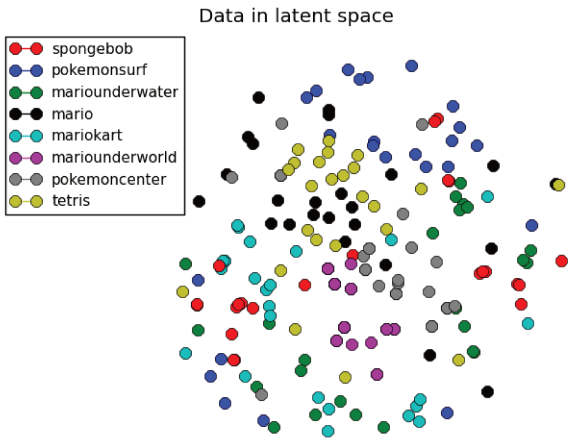


Figure 16: Visualization of the latent space of a VAE trained on MIDI sequences from video game songs. Reproduced from [47].

However, I believe that the dataset used in [47] appears to be very small: the generated samples[1] tend to only mix between the original sequences. Furthermore, there is no way to determine any specific "meaningful" direction in the latent space. This means that moving in this latent space can only be made in a random way, which is not necessarily a feature that we want when devising tools for interactive composition. These problems are addressed in Chap. 9 where I propose a novel way to encode additional information in the latent space so that we can exploit its meaningful structure to generate variations of a sequence in an intended way. This model has been applied in a similar way in [149] on a bigger dataset of MIDI files. The interesting difference in their approach is that the authors also condition the encoding and decoding RNNs on the MIDI tracks meta information such as the genre or the main instrument type. The generated sequences are coherent but could hardly be identified as belonging to a precise style.

1 Generations from [47] can be heard at http://youtu.be/cu1_uJ9qkHA

Part II

POLYPHONIC MUSIC MODELING

5

STYLE IMITATION AND CHORD INVENTION IN POLYPHONIC MUSIC WITH EXPONENTIAL FAMILIES

INTRODUCTION

Polyphonic tonal music is often considered as a highlight of Western civilization. As discussed in Ch. 2.2, today's music is still largely based on complex structures invented and developed since the Renaissance period, and modeled, e.g. by Jean Philippe Rameau [126] in the XVIIth century. In particular, polyphonic music is characterized by an intricate interplay between melody (single-voice stream of notes) and harmony (progression of simultaneously-heard notes). Additionally, composers tend to develop a specific *style*, that influences the way notes are combined together to form a musical piece.

We presented in Ch. 4 an overview on the recent usages of deep learning techniques for symbolic music generation, discussing both about models for both monophonic and polyphonic models. In this part, we will focus on polyphonic music since it requires a very specific treatment and raises original problems (see Sect. 2.2.1).

Contrary to the intuition, the first generative models on music were devised for polyphonic music and not monophonic music.

This started in the 50s with the Illiac Suite [94], which used Markov chains to produce 4-voice music, controlled by hand-made rules. Since then, many models for polyphonic music generation have been proposed.

In this chapter we address the issue of learning from a homogeneous dataset of polyphonic music, with the aim of producing new musical pieces that satisfy additional user constraints. The model that we propose is a simple model, which cannot be considered as a deep learning model. Its advantage is its precise mathematical formulation and the sampling procedure that it introduces. We will see how to improve this model in Ch. 6 by introducing deep neural networks for predictions. If the results are by far better with the deep network-based model, we will see that the price to pay in order to obtain these improved results is that we loose theoretical guarantees about the convergence of our sampling procedure.

EXISTING APPROACHES ON POLYPHONIC MUSIC GENERATION

In practice, an interesting model for polyphonic music generation should satisfy three requirements: statistical *accuracy* (capturing faith-

fully statistics of correlations at various ranges, horizontally and vertically), *flexibility* (coping with arbitrary user constraints), and *generalization* capacity (inventing new material, while staying in the style of the training corpus).

Models proposed so far fail on at least one of these requirements. In [46], the authors propose a chord invention framework. However, this framework is not based on agnostic learning, and requires a handmade ontology. The approach described in [117] consists in a dynamic programming template enriched by constrained Markov chains. This approach generates musically convincing results [116] but is *ad hoc* and specialized for jazz. Furthermore it cannot invent any new voicing (the vertical ordering of the notes in a chord) by construction. In [66] and [4], the authors describe an approach using Hidden Markov Models (HMMs) trained on an annotated corpus. This model imitates the style of Bach chorales and the authors report good cross entropy measures. However, the described model is also not able to produce new voicings but can only replicate ones that are found in the training corpus. Another related approach is [76], which uses HMMs on specific hand-crafted chord representations to generate homorhythmic sequences. These representations are based on an expert knowledge of the common-practice harmony and are called General Chord Type (GCT) [22]. A drawback of these models is that they are not agnostic, in the sense that they include a priori knowledge about music such as the concept of dissonance, consonance, tonality or scale degrees.

Agnostic neural-network based approaches have been investigated with promising results. We presented the architectures as well as the pros and cons of these models in Sect. 4.1.2.1 and we refer the reader to this section. In short, the drawbacks of these models are that they require large and coherent training sets which are not always available. More importantly, how to enforce additional user constraints (flexibility) it is not clear and their invention capacity is not demonstrated.

In this chapter we introduce a graphical model based on the maximum entropy principle for learning and generating polyphonic music. Such models have been used for music retrieval applications [123], but never, to our knowledge, for polyphonic music generation. This model requires no expert knowledge about music and can be trained on small corpora. Moreover, generation is extremely fast.

We show that this model can capture and reproduce pairwise statistics at possibly long range, both horizontally and vertically. These pairwise statistics are also able, to some extent, to capture implicitly higher order correlations, such as the structure of 4-note chords. The model is flexible, as it allows the user to post arbitrary unary constraints on any voice. We also show that this model exhibits a remarkable capacity to invent new but "correct" chords. In particular

we show that it produces harmonically consistent sequences using chords which did not appear in the original corpus.

In Sect. 5.3 we present the model for n-parts polyphony generation. In Sect. 5.4.2, we report experimental results about chord invention. In Section 5.4.4 we discuss a range of interactive applications in music generation. Finally, we discuss how the "musical interest" of the generated sequences depends on the choice of our model's hyperparameters in Sect. 5.4.5.

THE MODEL

The model we propose is based on the maximum entropy model for monophonic music generation that is described in [134]. The latter is extended to handle several voices instead of one. In order to do so, we introduce a graphical model on a grid-like representation of the music which models

- horizontal interactions (interactions between two notes belonging to the same voice),
- vertical interactions (interactions between two notes played at the same time but belonging to different voices)
- diagonal interactions (interactions between notes played at different time steps and different voices).

We formulate this model as a discrete exponential family obtained by a product of experts (one for each voice).

Description of the model

We aim to learn sequences s of n-part chord sequences. A sequence $s = [c_1, \dots, c_l]$ is composed of l chords where the j^{th} chord is denoted by

$$c_j := [s_{1j}, s_{2j}, \dots, s_{nj}],$$

where the note s_{ij} is an integer pitch belonging to the pitch range $\mathcal{A}_i \subset \mathbf{N}$. The ith part or *voice* corresponds to

$$v_i := [s_{i1}, s_{i2}, \dots, s_{il}].$$

Our model is based on the idea that chord progressions can be generated by replicating the occurrences of pairs of neighboring notes. In order to generate sequences of any size, we suppose that our model is invariant by translation in time: its aim is more to capture the local "texture" of the chord sequences than to capture long-range correlations. It is worth noting that similar ideas have been shown to be

successful in modeling highly combinatorial and arbitrary structures such as English four-letter words [142].

We denote by K the model *scope*, which means that we consider that chords distant by more than K time steps are conditionally independent given all other variables. We focus on the interaction between neighboring notes and try to replicate the co-occurrences between notes. A natural way to formalize this is to introduce a family of functions (or features) such that each member of this family counts the number of occurrences of a given pair of notes. As a result, the finite number of features we want to learn can be written as a family

$$\left\{ f_{ab,ijk} \quad \text{s.t.} \quad \begin{array}{l} a \in \mathcal{A}_i, \quad b \in \mathcal{A}_j \\ i,j \in [1,n], \quad k \in [-K,K] \end{array} \right\} \tag{12}$$

of functions over chord sequences, where

$$f_{ab,ijk}(s) := \sharp \left\{ m \quad \text{s.t.} \quad s_{i,m} = a \text{ and } s_{j,m+k} = b \right\} \tag{13}$$

stands for the number of occurrences of pairs of notes (a,b) in the chord sequence s such that note a at voice i is played k time steps before note b at voice j. We can represent this family of *binary connections* as a graphical model as can be seen in Fig. 17. On this illustration, each subfamily

$$\{ f_{ab,ijk}, \quad \forall a \in \mathcal{A}_i, \quad b \in \mathcal{A}_j \}$$

is represented by a link between two notes. Our model has also *unary* parameters, acting on single notes and modeling the single notes marginal distributions. For notational convenience we will treat these unary parameters as binary connections between pairs of identical notes (connections such that a=b, i=j, k=0) and call them *local fields* after the corresponding statistical physics terminology [103]. We differentiate four types of connections: unary (local fields) and binary "horizontal", "vertical" and "diagonal" connections. We implicitly identify $f_{ab,ijk}$ with $f_{ba,ji(-k)}$, for all $k \in [-K,K]$, $i,j \in [1,n]$.

From now on, we will denote the set of indexes of the family $\{f_{ab,ijk}\}$ by $\mathcal{P}$. We note that there is approximately

$$n^2(2K+1)\,|\mathcal{A}|^2 \tag{14}$$

indexes, where $|\mathcal{A}|$ stands for the mean alphabet size

$$|\mathcal{A}| = \frac{1}{n}\sum_{i=1}^{n} |\mathcal{A}_i|.$$

Using only a subset of the family $\{f_{ab,ijk}\}$ given by (12) can reduce the number of parameters while leading to good results. Indeed, if we consider that notes in different voices are conditionally independent

if they are distant by more than $L \leqslant K$ time steps, we obtain an index set of size approximately equal to

$$(n \times K + \frac{n^2 - n}{2} L)|\mathcal{A}|^2.$$

In the following, $\mathcal{P}$ can designate the whole set of indexes within scope K as well as any of its subset.

Let $\{\mu^{ab,ijk}\}$ for $(ab, ijk) \in \mathcal{P}$ be real numbers. From all distributions $P(s)$ over sequences such that the averages over all possible sequences of length l verify

$$\sum_s P(s) f_{ab,ijk}(s) = \mu^{ab,ijk}, \quad \forall (ab, ijk) \in \mathcal{P}, \tag{15}$$

it is known that the exponential distribution with statistics $\{f_{ab,ijk}\}$ and parameters $\{\mu^{ab,ijk}\}$ is the one of maximum entropy (i.e. the one with the greatest "uncertainty").

We consider an energy-based model of parameter

$$\theta := \left\{\theta^{ab,ijk} \in \mathbf{R}, \quad \forall (ab, ijk) \in \mathcal{P}\right\}$$

given by

$$P(s|\theta) = \frac{e^{-E(s,\theta)}}{Z(\theta)}, \tag{16}$$

where

$$E(s, \theta) := -\sum_{ab,ijk} \theta^{ab,ijk} f_{ab,ijk}(s) \tag{17}$$

is usually called the *energy* of the sequence s, and where

$$Z(\theta) := \log(\sum_s e^{-E(s,\theta)})$$

is called the *normalizer* or the *partition function* (the function such that P defines a probability function over sequences of size l). The sum in the partition function is for every s of size l. There are approximately $|A|^l$ such sequences, which makes the exact computation of the partition function intractable in general.

Training

We consider a training dataset $\mathcal{D}$ composed of N n-part sequences $s^{(1)}, \ldots, s^{(N)}$. In the following, we suppose for clarity that all these sequences are concatenated into one long sequence s, so that we can drop the exponents. Since we are dealing with discrete data, gradient techniques such as score matching [72] cannot be used. Instead, we choose to minimize the negative *pseudo-log-likelihood* of the data

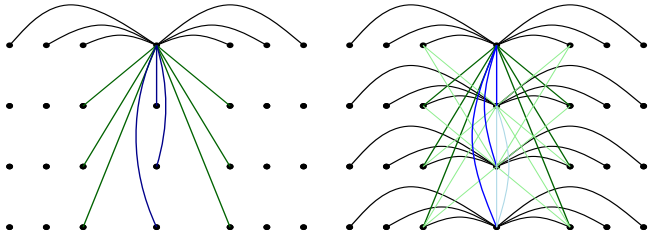


Figure 17: Example of binary connections involving the first voice's fourth note (on the left) and the fourth chord (on the right). Horizontal connections are in black, vertical in green and diagonal in blue.

[45, 127] in order to find an approximation of the true maximum likelihood estimator. It consists in approximating the negative log-likelihood function

$$\mathcal{L}(\theta, s) = -\log P(s|\theta) \tag{18}$$

by the mean of conditional log-likelihoods of a note given the others. That is

$$\mathcal{L}(\theta, s) \approx -\frac{1}{nl} \sum_{ij} \log P(s_{ij}|s_{\setminus s_{ij}}, \theta), \tag{19}$$

where $s_{\setminus s_{ij}}$ denotes all notes in s except s_{ij}. The conditional probabilities are calculated as

$$P(s_{ij}|s_{\setminus s_{ij}}, \theta) = \frac{P(s, \theta)}{\sum\limits_{\substack{c \in \mathcal{A}_i \\ s'_{\setminus s'_{ij}} = s_{\setminus s_{ij}} \\ s'_{ij} = c}} P(s', \theta)} \tag{20}$$

where the sum in the denominator is on chord sequences s' equal to s except for the note in position ij.

Due to the particular structure of the probability density function (16) and the choice of the statistics (12), we note that we can write

$$P(s_{ij}|s_{\setminus s_{ij}}, \theta) = P(s_{ij}|\mathcal{N}_K(i, j, s), \theta),$$

where $\mathcal{N}_K(i, j, s)$ stands for the neighbors of note s_{ij} in s that are at a distance inferior to K time steps. We now express our dataset $\mathcal{D}$ as a set of samples $(x, \mathcal{N})$ consisting of a note x together with its K-distant neighbors, ignoring border terms whose effect is negligible. More precisely, we write the dataset

$$\mathcal{D} = \{\left(s_{ij}, \mathcal{N}_K(i, j, s)\right), \quad \forall i \in [1, n], \quad \forall j \in [K+1, l-K-1]\} \tag{21}$$

and split it into n datasets $\mathcal{D}_i$ such that

$$\mathcal{D}_i = \left\{\left(s_{ij}, \mathcal{N}_K(i, j, s)\right), \quad \forall j \in [K+1, l-K-1]\right\}.$$

Each element of this dataset consists in a pair of an input $\mathcal{N}$ (the neighborhood of a given note) and a label $y \in \mathcal{A}_i$ where $i \in [1, n]$. Those notations set, we can rewrite Eq. 19 as

$$\begin{aligned}\mathcal{L}(\theta, \mathcal{D}) &\approx -\frac{1}{\sharp\mathcal{D}} \sum_{(y,\mathcal{N})\in\mathcal{D}} \log P(y|\mathcal{N}, \theta) \\ &:= \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_i(\theta, \mathcal{D}_i),\end{aligned} \tag{22}$$

where

$$\mathcal{L}_i(\theta, \mathcal{D}_i) = -\frac{1}{\sharp\mathcal{D}_i} \sum_{(y,\mathcal{N})\in\mathcal{D}_i} \log P(y|\mathcal{N}, \theta)$$

is the negative conditional log-likelihood function for voice i. This consists in minimizing the mean of n negative log-likelihood functions $\mathcal{L}_i$ (one for each voice) over the data $\mathcal{D}_i$. This method has the advantage of being tractable since there are only $\sharp\mathcal{A}_i$ terms in the denominator of Eq. 20 and lead to good estimates [8]. This can be seen as the likelihood of a product of experts using n modified copies (featuring also vertical and diagonal connections) of the model presented in [134].

We need to find the parameters that minimize the sum of n convex functions. Computing the gradient of $\mathcal{L}_i$ for $i \in [1, n]$ with respect to any parameter θ_* with $* \in \mathcal{P}$ gives us

$$\begin{aligned}&\frac{\partial\mathcal{L}_i(\theta, \mathcal{D}_i)}{\partial\theta_*} = \\ &\frac{1}{\sharp\mathcal{D}_i} \sum_{\substack{s=(y,\mathcal{N}) \\ (y,\mathcal{N})\in\mathcal{D}_i}} \left(\frac{f_*(s)e^{-E(s,\theta)}}{\sum_{\substack{s':=(y',\mathcal{N}) \\ y'\in\mathcal{A}_i}} f_*(s')e^{-E(s',\theta)}} - f_*(s) \right).\end{aligned} \tag{23}$$

This can be written as

$$\frac{\partial\mathcal{L}_i(\theta, \mathcal{D}_i)}{\partial\theta_*} = \langle f_* \rangle_{P(.|\mathcal{N},\theta)} - \langle f_* \rangle_{\mathcal{D}_i}$$

which is the difference between the average value of f_* taken with respect to the conditional distribution (20) and its empirical value.

A preprocessing of the corpus is introduced in order to efficiently compute the gradient sums.

Finally, the function $g(\theta)$ that we optimize is the L1-regularized version of $\mathcal{L}(\theta, \mathcal{D})$ with regularization parameter λ. Concretely, this means that we consider as our objective function

$$g(\theta) = \mathcal{L}(\theta, \mathcal{D}) + \lambda\|\theta\|_1, \tag{24}$$

where $\|.\|_1$ is the usual L1-norm, which is the sum of the absolute values of the coordinates of the parameter. This is known as the Lasso regularization whose effects (overfitting reduction, sparsity) are widely discussed throughout statistical learning literature [50].

Generation

Generation is performed using the Metropolis-Hastings algorithm, which is an extensively used sampling algorithm (see [27] for an introduction). Its main feature is the possibility to sample from an unnormalized distribution since it only requires to compute ratios of probabilities

$$\alpha := \frac{P(s', \theta)}{P(s, \theta)} \tag{25}$$

between two sequences s' and s.

It is an iterative algorithm that starts from a random sequence s and iteratively modifies it. The main loop of this algorithm is as follow: we draw a sequence proposal s' (which depends on the current sequence s) and compute the α ratio (25. We then accept s' as our current sequence with probability $\min(\alpha, 1)$ or reject this proposal with probability $1 - \min(\alpha, 1)$ and keep s as our current sequence. We are assured that the sequences obtained are distributed according the objective distribution $P(.|\theta)$ after a sufficient number of iterations of this procedure i.e. once we have attained the mixing time of the Monte Carlo Markov Chain (MCMC) described above. We chose as our proposal distribution for s' to draw s' uniformly among all sequences that differ by only one note from s.

By slightly modifying the proposal distribution on s', we can use this algorithm to enforce unary constraints on the produced sequences. Indeed, if we only propose sequences s' that contains a sequence $\{n_{ij}\}_{(ij)\in\mathcal{C}}$ of imposed notes, i.e. such that

$$s_{ij} = n_{ij}, \quad \forall (ij) \in \mathcal{C},$$

where $\mathcal{C}$ contains the indexes of the constrained notes, the Metropolis-Hastings algorithm samples from the distribution

$$P(s|s_{ij} = n_{ij}), \quad \forall (ij) \in \mathcal{C}.$$

This enables us, for instance, to provide reharmonizations of a given melody. Other types of constraints are possible: we can, for example, add pitch range constraints on given notes, which means that we are given a set

$$\left\{\mathcal{A}_{ij} \subset \mathcal{A}_i, \quad \forall (ij) \in \mathcal{C}\right\}$$

such that we only propose sequences s' where

$$s_{ij} \in \mathcal{A}_{ij}, \quad \forall (ij) \in \mathcal{C}.$$

A musical application of this constraint is that we can impose a chord label without imposing its voicing.

EXPERIMENTAL RESULTS

We report experiments made using a set of 343 four-voice ($n = 4$) chorale harmonizations by Johann Sebastian Bach [10]. In order to evaluate the chord creation capabilities, we only retained the notes that are heard on beats. Sect.5.4.6 shows how we can easily produce rhythm using our model.

We transposed every chorale in the key of C and considered 2 corpora: a corpus with chorales in a major key and a corpus with chorales in a minor key.

The quadratic number of parameters (thanks to the exclusive use of binary connections) makes the learning phase computationally tractable.

We used the L-BFGS method from Stanford CoreNLP optimization package [100] to perform the gradient descent.

In the next sections, we report on the model's accuracy (its style imitation capacity), invention capacity, and flexibility.

Style imitation

We investigated the capabilities of the proposed model to reproduce pair-wise correlations of the training set. Figure 18 shows a scatter plot comparing the (normalized) values of each binary connection $f_{ab,ijk}$ in the generated sequences versus the ones of the original training corpus. The model was trained on a corpus of 51 major chorales, which represents the equivalent of a 3244-beat long chord sequence. We chose to differentiate horizontal connections from vertical and diagonal ones by introducing a parameter L as mentioned in Sect. 5.3.1. We took $K = 4$, $L = 2$, $\lambda = 3e^{-5}$ as parameters and generated a 100000-beat long sequence. For a discussion on the choice of the regularization parameter, we refer the reader to Sect. 5.4.5. We see that despite the small amount of data, the alignment between the generated pair occurrences and the original ones is quite convincing.

The generation procedure needs solely to compute the ratios (25), which can be done in approximately $O(nK)$ operations. Indeed, since the sequences differ by only one note, only contributions of its neighboring notes have to be taken into account. This has to be compared with the approximate number of parameters (14). Experimentally, we find that the number of metropolis steps to achieve convergence is of order $O(n\mathcal{A})$ which enables these models to be used in real-time applications.

We argue that this model does not only reproduce pairwise statistics but can also capture higher-order interactions, which makes it suitable for style imitation. Indeed, the way that the binary connections are combined in Eq. 16 makes the model able to reproduce correct voicings. This is interesting knowing that the interaction between

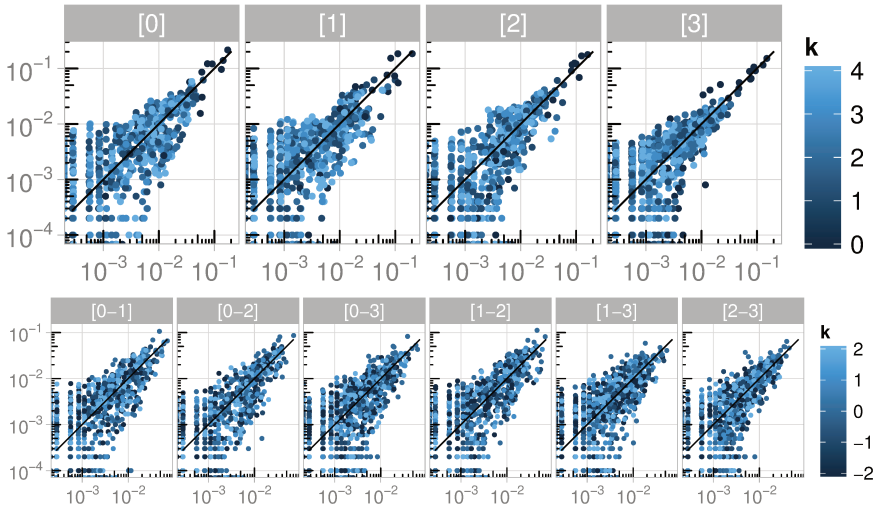


Figure 18: Generated correlations versus corpus correlations. We grouped binary connections involving the same voices. Colors correspond to the k parameter in (13).

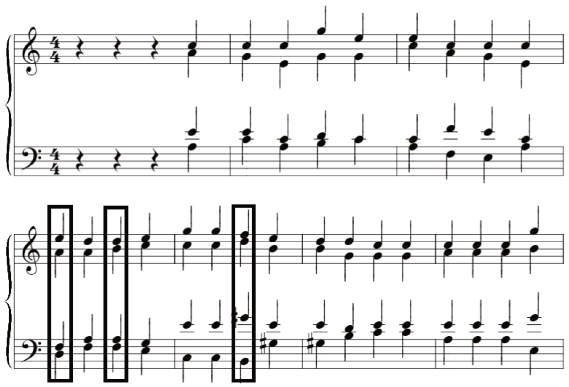


Figure 19: Reharmonization of a Bach chorale. "Invented" chords are in boxes.

the notes composing a chord is a conceptually an interaction of order n. Figure 19 shows chords with nice voicings: voices do not cross, triads have correct doubling and each separate voice has a coherent shape. A detailed analysis of the chord creation capacity of the system is made in Sec. 5.4.2. We discuss the capability of the system to reproduce other higher-order patterns in Sect. 5.4.3.

Chord Invention

We claim that even if our model only takes into account horizontal, vertical and diagonal correlations between two notes, it is able to generate new chords in the learned "style". Three categories of generated chords can be distinguished: the *cited* chords which appear in the model's training set, the *discovered* chords which do not appear in the training set but can be found in other Bach's chorales and the

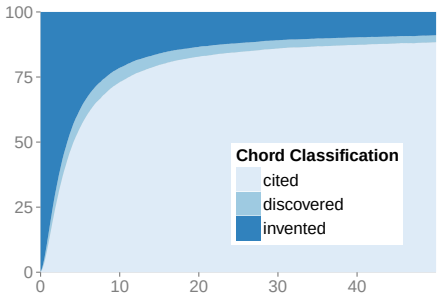


Figure 20: Evolution of the repartition (in %) between cited, discovered and invented chords during Metropolis-Hastings generation as a function of the normalized number of iterations).

invented chords which do not belong to any of the above categories. We used the same model as above to plot in Fig. 20 the repartition between the different categories of chords as a function of the number of Metropolis steps (divided by $|\mathcal{A}|nl$) during the Metropolis-Hastings generation. These curves highly depend on the model parameters and on the structure and the size of the corpus. Nonetheless we can note the characteristic time for the model to sample from the equilibrium distribution. For every parameter set we tested, we observed that when convergence is reached, the proportion of invented and discovered chords seems fixed and significant.

A closer investigation shows that most of these "invented" chords can in fact be classified as valid in the style of Bach by an expert. The majority of the invented chords is composed of "correct" voicings of minor or major triads, seventh chords and chords with non-chord tones. Fig. 19 exhibits interesting "inventions" such as an (unprepared) $9-8$ resolution, a dominant ninth and a diminished seventh. Other invented chords are discordant. A blindfolded evaluation was conducted to assess to which extent listeners are able to distinguish invented chords from cited ones. Three non professional music-loving adult listeners were presented with a series of invented chords extracted from generated sequences, and played with their context (i.e. 4 chords before and 4 chords after). They were asked whether the central chord was "good" or not. Results show that, in average, 75% of the invented chords were considered as acceptable.

Higher-order interactions

The same analysis as in Sec. 5.4.2 can be made for other structures than chords. We chose to investigate to which extent the model is able to reproduce the occurrences of *quadrilateral tuples*. For a sequence s, we define the *quadrilateral tuple* between voices i and i' at position j to be the tuple

$$(s_{i(j)}, s_{i(j+1)}, s_{i'(j)}, s_{i'(j+1)}).$$

Table 1: Percentage of cited/discovered/invented quadrilateral tuples

	cited	discovered	invented
$K = 4, L = 2$	61.5	9.6	28.9
$K = 0, L = 0$	24.1	7.8	63.1
independent	8.4	4.4	87.2

These tuples are of particular interest since many harmonic rules apply to them, e.g. such as the prohibition of consecutive fifths and consecutive octaves, often considered to be forbidden in counterpoint. Table. 1 compares the percentage of cited/discovered/invented quadrilateral tuples generated by different models. The models we considered are: the model of Sec. 5.4.1; a model containing only vertical interactions and an *independent* model which only reproduces pitch frequencies.

This table shows that an important part of those higher order structures is reproduced. However, analysis exhibits limitations on the higher-order statistics that can be captured (see for instance Fig. 19). Indeed, even if our preprocessed corpus contains some of these "rules violations", our model is unable to statistically reproduce the number of such structures (they are 2 to 10 times more frequent than in the original corpus). We discuss non agnostic methods that can integrate these particular rules in Sec. 5.5.

Flexibility

As claimed in Sec. 5.3.3, we can use our model to generate new harmonizations of a melody. Indeed, the simplicity and adaptability of this model allows it to be "twisted" in order to enforce unary constraints while still generating sequences in the learned style. As our model is in a specified key (all chorales were transposed in the same key), we can thus provide convincing Bach-like harmonizations of plainsong melodies provided they "fit" in the training key. Fig. 21 shows two reharmonizations of Beethoven's Ode to Joy with different unary constraints[1]. It is worth noting that even if we put constraints on isolated notes, and not on full chords, the constraints propagate well both vertically (the voicings are correct) and horizontally (the progression of chords around the constrained notes is coherent). This opens up a wide range of applications. Those examples show how enforcing simple unary constraints can be used to produce interesting musical phenomena during reharmonization such as:

- original harmonies (see for instance the 1^{st}, 2^{nd} and 4^{th} constraints in Fig. 21)

1 Music examples can be heard on the http://flowmachines.jimdo.com/ website

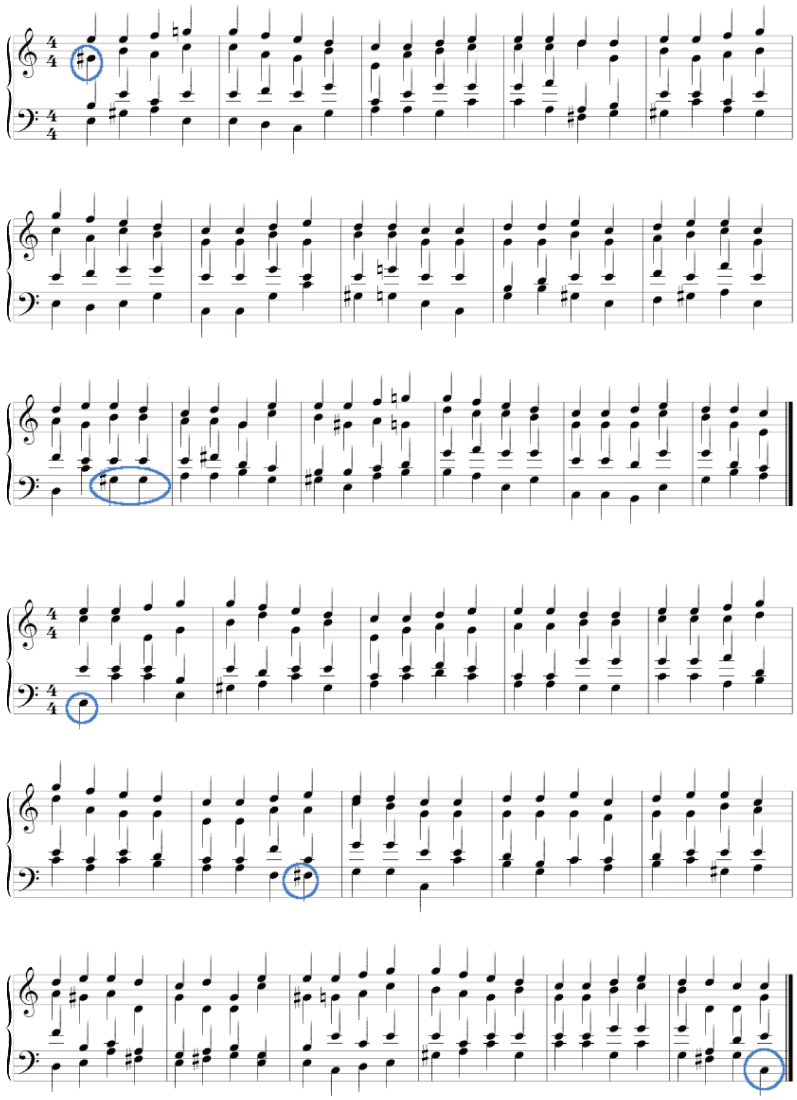


Figure 21: Two reharmonizations of Beethoven's Ode to Joy with constraints. Constrained notes are circled.

- expected harmonies (3^{rd} constraint in Fig. 21)
- a sense of long-range correlation by creating cadenzas (last constraint in Fig. 21).

We can modify the constrained Metropolis-Hastings sampling scheme of Sect. 5.3.3 so that we can harmonize any melody

$$v_0 = [s_{01}, s_{02}, \ldots, s_{0l}]$$

of size l, even if this melody modulates and does not stay in the key of the training set. For each beat $j \in [1, l]$, we use a melody analyzer to yield the current key k_j at beat j. If the new proposed sequence s' differs from the current one s by a note at beat j, we compute the acceptance ratio (25) by using the probability distribution of the model

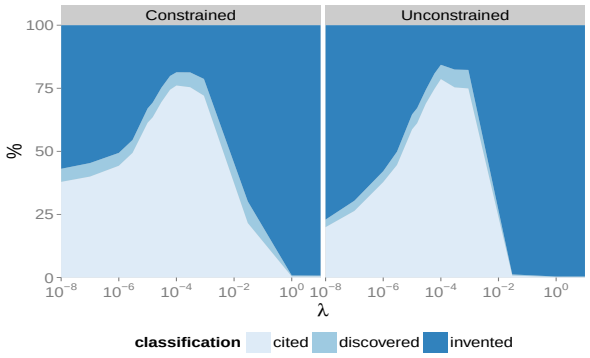


Figure 22: Cited/discovered/invented repartition at equilibrium as a function of λ in the unconstrained and constrained cases.

trained in key k_j. By doing so, we choose the appropriate model for each chunk of the melody and "glue" the results together seamlessly.

Impact of the regularization parameter

In this section we discuss the choice of the regularization parameter λ of Eq. 24. The benefits of introducing a L1-regularization are multiple: it makes the loss function (24) strictly convex (in our case, we do not need to determine if the family (12) is a sufficient family), tends to obtain sparse parameters and prevents overfitting As our model possesses an important number of parameters compared to the number of samples, adding a regularization term during the training phase appears to be mandatory for obtaining good results in the applications we mentioned in Sect. 5.3.3 and Sect. 5.4.4.

We evaluate the impact of the choice of λ on the cited/discovered/invented classification curves (Fig. 19). We compare the mean repartition of chords in the unconstrained generation case and in the reharmonization case, in which the first voice is constrained. The training corpus used is the same as the one used in Sect. 5.3.3, with $K = 4$, $L = 2$ and varying λ . We use the first voice of the chorales from the testing corpus as constraints. Results are presented in Fig. 22.

A clear influence of λ appears for both the unconstrained and constrained generations. However, these curves are not sharply peaked and it seems not clear which regularization parameter could be the most musically-interesting one.

In order to answer this question, we investigated to which extent the regularization parameter influences the model's ability to reproduce a wide variety of chords, chords that are either seen in the training set or rediscovered in the testing set.

We introduce two quantities revealing the diversity of the generated sequences: the percentage of *restitution* of the training corpus (the number of cited chords counted without repetition and normal-

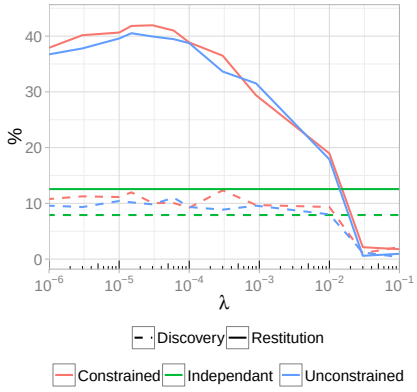


Figure 23: Plot of the restitution and discovery percentages as a function of λ in the unconstrained and constrained cases.

ized by the total number of different chords in the training corpus) and the percentage of *discovery* of the testing corpus (the number of different discovered chords, normalized by the number of chords in the testing corpus which are not in the training corpus and counted without repetition). The evolution of the restitution/discovery percentages as a function of λ is plotted in Fig. 23 for both the constrained and unconstrained generation cases.

Both figures exhibit the same behavior. High values of λ lead to uniform models and low values of λ lead to models which overfit the training data. But the most interesting observation is that their maximum is not attained for the same value of λ revealing the possibility to control the trade-off between the invention capacity, the diversity and the faithfulness with respect to the training corpus in the generated sequences.

Rhythm

We focused in this work on chord reproduction and invention. Since the model is by construction pairwise and chords are, in our examples, four-notes objects, it is in itself a non-trivial question to assess the capacity of such a model to accurately model the style of the corpus. In order to simplify the analysis on four-notes chords, we thus chose to work on homorhythmic sequences and we discarded all notes from the training corpus that do not fall on the beat. However, real music is not necessarily homorhythmic: notes have varying durations and form temporal patterns which take place with respect to some periodic pulse on a temporal canvas.

We propose a simple way to extend our model in order to account for rhythmic patterns. The model initially presented in [134] and extended in 5.3 is translation invariant. For rhythmic patterns to emerge, we need to break this translation invariance. We do so by introducing

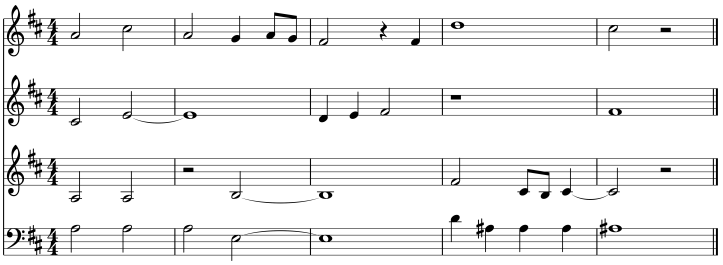


Figure 24: Generated polyphonic music with rhythm. The original corpus on which the model is trained is *Missa Sanctorum Meritis, Credo* by Giovanni Pierluigi da Palestrina.

position-dependent parameters. More specifically, we choose a cycle which is repeated over time and within which the translation invariance is broken. Such cycle can be for example one or two bars of music. We then divide this cycle in equal time bins which correspond to all possible positions where a note can start or end. We call these time bins *metrical positions*. We could then define different parameters between notes having different metrical positions. However, that would lead to a very large number of parameters and would lead to a very inaccurate learning (it can be argued that the number of parameters must be smaller than the number of data points). We have found that a good compromise is to let the unary parameters (local fields) be position-dependent while keeping the translation invariance for the true binary parameters. This leads to a negligible increase in the number of parameters since the unary parameters are of order $|\mathcal{A}|$ whereas the binary ones are of order $|\mathcal{A}|^2$. Finally, in order to obtain a variety of note durations as well as rests, we introduce two additional symbols in the alphabet. One symbol for rests and one symbol that signifies the continuation of the previous note in the current metrical position.

The above procedure has the following effect: the position-dependent parameters are biasing locally the occurrence of symbols (pitches, rests or continuations of the previous pitch) in a way that is consistent with the original corpus. This leads to the emergence of rhythmic patterns of the same kind as the ones found in the corpus. An example can be seen in 24. In order to generate this example, we used a cycle of one bar and divided it in 8 equal parts. This subdivision corresponds to considering eighth notes (quavers) as the smallest duration, which is also the smallest division found in the corpus (here *Missa Sanctorum Meritis, Credo* by Giovanni Pierluigi da Palestrina).

DISCUSSION AND FUTURE WORK

We proposed a probabilistic model for chord sequences that captures pairwise dependencies between neighboring notes. The model is able to reproduce harmonic progressions without any prior information and invent new "stylistically correct" chords. The possibility to sam-

ple with arbitrary unary user-defined constraints makes this model applicable in a wide range of situations. We focused mainly on the chord creation and restitution capabilities which is, from our point of view, its most interesting feature, an analysis of the plagiarism of monophonic graphical models being made in [134]. We showed that even if the original training set is highly combinatorial, these probabilistic methods behave impressively well even if high-order hard constraints such as parallel fifths of octaves cannot be captured. This method is general and applies to all discrete n-tuple sequences. Indeed, we used as features the occurrences of notes, but any other family of functions could be selected. For instance, adding occurrences of parallel fifths or parallel octaves in the family (12) would be possible and would only require $O(n^2|\mathcal{A}|^2)$ parameters, which does not increase our model complexity.

The utmost importance of the regularization parameter suggests to investigate finer and more problem-dependent regularizations such as group lasso [51] or other hierarchical sparsity-inducing norms [9]. We believe that having more than a single scalar regularization parameter λ can lead to a better control of the "creativity" of our model.

6

DEEPBACH: A STEERABLE MODEL FOR BACH CHORALES GENERATION

INTRODUCTION

The composition of polyphonic chorale music in the style of J.S. Bach has represented a major challenge in automatic music composition over the last decades. We refer the reader to Sect. 2.3.4 for a presentation of this dataset.

In the preceding chapter, we introduced a statistical for the generation of multi-part music and studied its generation properties. The results are promising on the corpus of J.S. Bach chorales, but the generated outputs cannot be confounded with original compositions by Bach. In this chapter, we will focus exclusively on the corpus of the chorale harmonizations by Johann Sebastian Bach. The aim is to obtain a generative model that can learn the style of the chorale harmonizations and generate high-quality chorales that could be considered as "composed by J.S. Bach", even by experts.

We will not lose sight of the importance of inventing new ways to interact with this generative model (as evoked in Sect. 3.4). The differences with the model introduced in Ch. 5 are notable, but links between the two methods exist. We believe that the transition from the shallow model of Sect. 5.3, which is more interpretative but generates less convincing samples, to the deep model of Sect. 6.2.2, which is less interpretative but generates state-of-the-art Bach-like chorales, worth noting.

As discussed in Sect. 2.3.4, this corpus of harmonizations is remarkable by its homogeneity and its size (389 chorales in [10], 402 in the music21 [38] Python package). Each of these short pieces are composed following similar principles. This implies characteristic rhythms, variety of harmonic ideas as well as characteristic melodic movements and makes the style of these chorale compositions easily distinguishable, even for non experts.

The difficulty, from a compositional point of view comes from the intricate interplay between harmony (notes sounding at the same time) and voice movements (how a single voice evolves through time). Furthermore, each voice has its own "style" and its own coherence. Finding a chorale-like reharmonization which combines Bach-like harmonic progressions with musically interesting melodic movements is a problem which often takes years of practice for musicians.

From the point of view of automatic music generation, the first solution to this apparently highly combinatorial problem was pro-

posed by [44] in 1988. This problem is seen as a constraint satisfaction problem, where the system must fulfill numerous hand-crafted constraints characterizing the style of Bach. It is a rule-based expert system which contains no less than 300 rules and tries to reharmonize a given melody with a generate-and-test method and intelligent backtracking. Among the short examples presented at the end of the paper, some are flawless. The drawbacks of this method are, as stated by the author, the considerable effort to generate the rule base and the fact that the harmonizations produced "do not sound like Bach, except for occasional Bachian patterns and cadence formulas." In our opinion, the requirement of an expert knowledge implies a lot of subjective choices.

A neural-network-based solution was later developed by [66]. This method relies on several neural networks, each one trained for solving a specific task: a harmonic skeleton is first computed then refined and ornamented. A similar approach is adopted in [4], but uses Hidden Markov Models (HMMs) instead of neural networks. Chords are represented as lists of intervals and form the states of the Markov models. These approaches produce interesting results even if they both use expert knowledge and bias the generation by imposing their compositional process. In [159, 160], authors elaborate on these methods by introducing multiple viewpoints and variations on the sampling method (generated sequences which violate "rules of harmony" are put aside for instance). However, this approach does not produce a convincing chorale-like texture, rhythmically as well as harmonically. Furthermore the resort to hand-crafted criteria to assess the quality of the generated sequences might rule out many musically-interesting solutions.

Recently, approaches which use neural networks and require no knowledge about harmony, Bach's style or music have been investigated with promising results. In [17] (developed in Sect. 4.1.2.1), chords are modeled with Restricted Boltzmann Machines (RBMs). Their temporal dependencies are learned using Recurrent Neural Networks (RNNs). Variations of these architectures based on Long Short-Term Memory (LSTM) units [69, 105] or GRUs (Gated Recurrent Units) have been developed by [95] and [30] respectively. However, these models which work on piano roll representations of the music are too general to capture the specificity of Bach chorales. Also, a major drawback is their lack of flexibility. Generation is performed from left to right. A user cannot interact with the system: it is impossible to do reharmonization for instance which is the essentially how the corpus of Bach chorales was composed. Moreover, their invention capacity and non-plagiarism abilities are not demonstrated.

A method that addresses the rigidity of sequential generation in music was first proposed in [134, 135] for monophonic music and later generalized to polyphony in the model described in Ch. 5. These

approaches advocate for the use of Gibbs sampling as a generation process in automatic music composition.

The most recent advances in chorale harmonization is arguably the BachBot model [93], a LSTM-based approach specifically designed to deal with Bach chorales. This approach relies on little musical knowledge (all chorales are transposed in a common key) and is able to produce high-quality chorale harmonizations. However, compared to our approach, this model is less general (produced chorales are all in the C key for instance) and less flexible (only the soprano can be fixed). Similarly to our work, the authors evaluate their model with an online Turing test to assess the efficiency of their model. They also take into account the fermata symbols (Fig. 7) which are indicators of the structure of the chorales.

In this chapter we introduce DeepBach, a dependency network [65] capable of producing musically convincing four-part chorales in the style of Bach by using a Gibbs-like sampling procedure. Contrary to models based on RNNs, we do not sample from left to right which allows us to enforce positional, unary user-defined constraints such as rhythm, notes, parts, chords and cadences. DeepBach is able to generate coherent musical phrases and provides, for instance, varied reharmonizations of melodies without plagiarism. Its core features are its speed, the possible interaction with users and the richness of harmonic ideas it proposes. Its efficiency opens up new ways of composing Bach-like chorales for non experts in an interactive manner similarly to what is proposed in [120] for leadsheets.

In Sect. 6.2 we present the DeepBach model for four-part chorale generation. We discuss in Sect. 6.3 the results of an experimental study we conducted to assess the quality of our model. Finally, we provide generated examples in Sect. 6.4.3 and elaborate on the possibilities offered by our interactive music composition editor in Sect. 6.4. All examples can be heard on the accompanying web page[1] and the code of our implementation is available on GitHub[2]. Even if our presentation focuses on Bach chorales, this model has been successfully applied to other styles and composers including Monteverdi five-voice madrigals to Palestrina masses.

DEEPBACH

We introduce a generative model which takes into account the distinction between voices. Sect. 6.2.1 presents the data representation we used. This representation is both fitted for our sampling procedure and more accurate than many data representation commonly used in automatic music composition. Sect. 6.2.2 presents the model's architecture and Sect. 6.2.3 our generation method. Finally, Sect. 6.2.4

1 https://sites.google.com/site/deepbachexamples/
2 https://github.com/Ghadjeres/DeepBach

provides implementation details and indicates how we preprocessed the corpus of Bach chorale harmonizations.

Data Representation

Notes and voices

We consider that only one note can be sung at a given time and discard chorales with voice divisions.

Since Bach chorales only contain simple time signatures, we discretize time with sixteenth notes, which means that each beat is subdivided into four equal parts. Since there is no smaller subdivision in Bach chorales, there is no loss of information in this process.

In this setting, a voice $\mathcal{V}_i = \{\mathcal{V}_i^t\}_t$ is a list of notes indexed by $t \in [T]$[3], where T is the duration piece (in sixteenth notes).

Rhythm

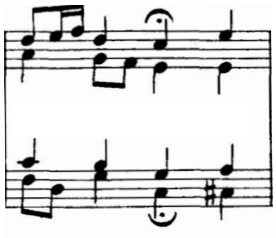
We use the melodico-rhythmic encoding described in Sect. 3.1.3. Namely, we choose to model rhythm by simply adding a *hold symbol* "__" coding whether or not the preceding note is held to the list of existing notes. This representation is thus unambiguous, compact and well-suited to our sampling method (see Sect. 6.2.3.4).

Metadata

The music sheet (Fig. 6b) conveys more information than only the notes played. We can cite:

- the lyrics,
- the key signature,
- the time signature,
- the beat index,
- an implicit metronome (on which subdivision of the beat the note is played),
- the fermata symbols (see Fig. 7),
- current key,
- current key signature,
- current mode (major/minor/dorian).

3 We adopt the standard notation $[N]$ to denote the set of integers $\{1, \dots, N\}$ for any integer N.



```
D5,__,E5,F5,D5,__,__,__,C5,__,__,__,E5
A4,__,__,__,G4,__,F4,__,E4,__,__,__,E4
C4,__,__,__,B3,__,__,__,G3,__,__,__,A3
F3,__,D3,__,G3,__,__,__,C2,__,__,__,C#2
 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1
 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0
```

(a) (b)

Figure 25: Extract from a Bach chorale and its representation as four voice lists and two metadata lists ($\mathcal{S}$ and $\mathcal{F}$). The hold symbol is displayed as "__" and considered as a note.

In the following, we will only take into account the fermata symbols, the subdivision indexes and the current key signature. To this end, we introduce:

- The *fermata* list $\mathcal{F}$ that indicates if there is a fermata symbol, see Fig. 7, over the current note, it is a Boolean value. If a fermata is placed over a note on the music sheet, we consider that it is active for all time indexes within the duration of the note.
- The *subdivision* list $\mathcal{S}$ that contains the subdivision indexes of the beat. It is an integer between 1 and 4: there is no distinction between beats in a bar so that our model is able to deal with chorales with three and four beats per measure.

Chorale

We represent a chorale as a couple

$$(\mathcal{V}, \mathcal{M}) \tag{26}$$

composed of voices and metadata. For Bach chorales, $\mathcal{V}$ is a list of 4 voices $\mathcal{V}_i$ for $i \in [4]$ (soprano, alto, tenor and bass) and $\mathcal{M}$ a collection of metadata lists ($\mathcal{F}$ and $\mathcal{S}$).

Our choices are very general and do not involve expert knowledge about harmony or scales but are only mere observations of the corpus. The list $\mathcal{S}$ acts as a metronome. The list $\mathcal{F}$ is added since fermatas in Bach chorales indicate the end of each musical phrase. The use of fermata to this end is a specificity of Bach chorales that we want to take advantage of.

Model Architecture

We choose to consider the metadata sequences in $\mathcal{M}$ as given. For clarity, we suppose in this section that our dataset is composed of

only one chorale written as in Eq. 32 of size T. We define a *dependency network* on the finite set of variables $\mathcal{V} = \{V_i^t\}$ by specifying a set of conditional probability distributions (parametrized by parameter $\theta_{i,t}$)

$$\left\{p_{i,t}(V_i^t | V_{\setminus i,t}, \mathcal{M}, \theta_{i,t})\right\}_{i\in[4],t\in[T]}, \tag{27}$$

where $\mathcal{V}_i^t$ indicates the note of voice i at time index t and $\mathcal{V}_{\setminus i,t}$ all variables in $\mathcal{V}$ except from the variable $\mathcal{V}_i^t$. As we want our model to be time invariant so that we can apply it to sequences of any size, we share the parameters between all conditional probability distributions on variables lying in the same voice, i.e.

$$\theta_i := \theta_{i,t}, \quad p_i := p_{i,t} \quad \forall t \in [T].$$

Finally, we fit each of these conditional probability distributions on the data by maximizing the log-likelihood. Due to weight sharing, this amounts to solving four classification problems of the form:

$$\max_{\theta_i} \sum_t \log p_i(\mathcal{V}_i^t | \mathcal{V}_{\setminus i,t}, \mathcal{M}, \theta_i), \qquad \text{for } i \in [4], \tag{28}$$

where the aim is to predict a note knowing the value of its neighboring notes, the subdivision of the beat it is on and the presence of fermatas. The advantage with this formulation is that each classifier has to make predictions within a small range of notes whose ranges correspond to the notes within the usual voice ranges (see 6.2.4).

For accurate predictions and in order to take into account the sequential aspect of the data, each classifier is modeled using four neural networks: two Deep Recurrent Neural Networks [122], one summing up *past* information and another summing up information coming from the *future* together with a non-recurrent neural network for notes occurring at the same time. Only the last output from the uppermost RNN layer is kept. These three outputs are then merged and passed as the input of a fourth neural network whose output is $p_i(\mathcal{V}_i^t | \mathcal{V}_{\setminus i,t}, \mathcal{M}, \theta)$. Figure 26 shows a graphical representation for one of these models. Details are provided in Sect. 6.2.4. These choices of architecture somehow match real compositional practice on Bach chorales. Indeed, when reharmonizing a given melody, it is often simpler to start from the cadence and write music "backwards."

Generation

Algorithm

Generation in dependency networks is performed using the pseudo-Gibbs sampling procedure. This Markov Chain Monte Carlo (MCMC) algorithm is described in Alg.1. It is similar to the classical Gibbs

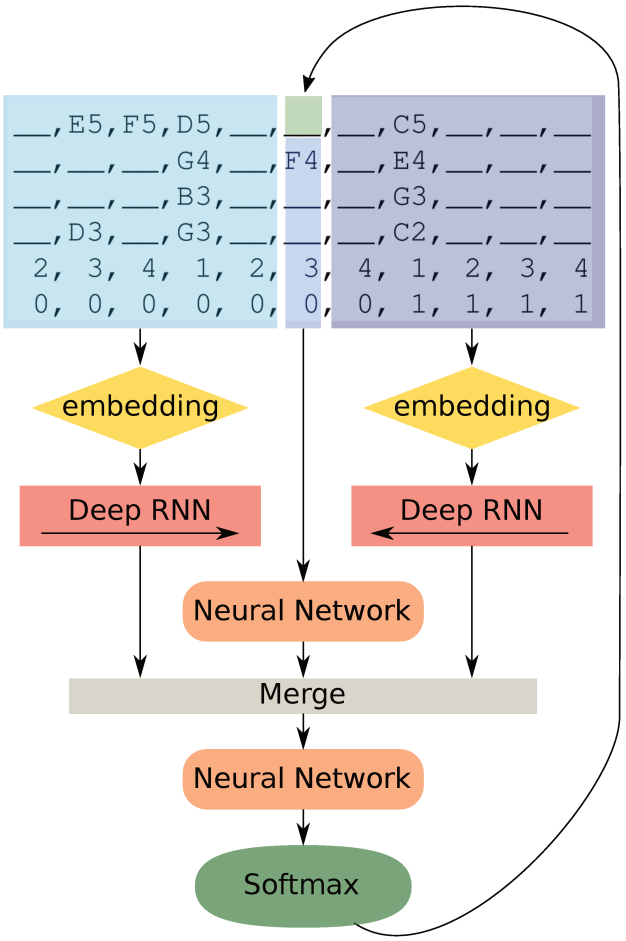


Figure 26: Graphical representations of DeepBach's neural network architecture for the soprano prediction p_1.

sampling procedure [52] on the difference that the conditional distributions are potentially incompatible [25]. This means that the conditional distributions of Eq. 27 do not necessarily comes from a joint distribution $p(\mathcal{V})$ and that the theoretical guarantees that the MCMC converges to this stationary joint distribution vanish. We experimentally verified that it was indeed the case by checking that the Markov Chain of Alg.1 violates Kolmogorov's criterion [77]: it is thus not reversible and cannot converge to a joint distribution whose conditional distributions match the ones used for sampling.

However, this Markov chain converges to another stationary distribution and applications on real data demonstrated that this method yielded accurate joint probabilities, especially when the inconsistent probability distributions are learned from data [65]. Furthermore, non reversible MCMC algorithms can in particular cases be better at sampling that reversible Markov Chains [155].

Algorithm 1 Pseudo-Gibbs sampling

1: **Input:** Chorale length L, metadata $\mathcal{M}$ containing lists of length L, probability distributions (p_1, p_2, p_3, p_4), maximum number of iterations M
2: Create four lists $\mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{V}_4)$ of length L
3: {The lists are initialized with random notes drawn from the ranges of the corresponding voices (sampled uniformly or from the marginal distributions of the notes)}
4: **for** m from 1 to M **do**
5: Choose voice i uniformly between 1 and 4
6: Choose time t uniformly between 1 and L
7: Re-sample $\mathcal{V}_i^t$ from $p_i(\mathcal{V}_i^t | \mathcal{V}_{\setminus i,t}, \mathcal{M}, \theta_i)$
8: **end for**
9: **Output:** $\mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{V}_4)$

Flexibility of the sampling procedure

The advantage of this method is that we can enforce user-defined constraints by tweaking Alg. 1:

- instead of choosing voice i from 1 to 4 we can choose to fix the soprano and only resample voices from 2, 3 and 4 in order to provide reharmonizations of the fixed melody
- we can choose the fermata list $\mathcal{F}$ in order to impose end of musical phrases at some places
- more generally, we can impose any metadata

- for any t and any i, we can fix specific subsets $\mathcal{R}_i^t$ of notes within the range of voice i. We then restrict ourselves to some specific chorales by re-sampling $\mathcal{V}_i^t$ from

  $$p_i(\mathcal{V}_i^t | \mathcal{V}_{\setminus i,t}, \mathcal{M}, \theta_i, \mathcal{V}_i^t \in \mathcal{R}_i^t)$$

  at step (5). This allows us for instance to fix rhythm (since the hold symbol is considered as a note), impose some chords in a soft manner or restrict the vocal ranges.

Performance

Note that it is possible to make generation faster by making parallel Gibbs updates on GPU. Steps (3) to (5) from Alg. 1 can be run simultaneously to provide significant speedups. Even if it is known that this approach is biased [41] (since we can update simultaneously variables which are not conditionally independent), we experimentally observed that for small batch sizes (16 or 32), DeepBach still generates samples of great musicality while running ten times faster than the sequential version. This allows DeepBach to generate chorales in a few seconds.

It is also possible to use the hard-disk-configurations generation algorithm (Alg.2.9 in [81]) to appropriately choose all the time indexes at which we resample so that:

- every time index is at distance at least δ from the other time indexes
- configurations of time indexes satisfying the relation above are equally sampled.

This trick allows to assert that we do not update simultaneously a variable and its local context and makes parallel updates possible and accurate.

Importance of the data representation

We emphasize on this section the importance of our particular choice of data representation with respect to our sampling procedure. The fact that we obtain great results using pseudo-Gibbs sampling relies exclusively on our choice to integrate the hold symbol into the list of notes.

Indeed, Gibbs sampling fails to sample the true joint distribution $p(\mathcal{V}|\mathcal{M}, \theta)$ when variables are highly correlated, creating isolated regions of high probability states in which the MCMC chain can be trapped. However, many data representations used in music modeling such as

- the piano roll representation,

- the couple (*pitch*, *articulation*) representation where *articulation* is a Boolean value indicating whether or not the note is played or held,

tend to make the musical data suffer from this drawback.

As an example, in the piano roll representation, a long note is represented as the repetition of the same value over many variables. In order to only change its pitch, one needs to change simultaneously a large number of variables (which is exponentially rare) while this is achievable with only one variable change with our representation. In Fig. 27 we show on a toy example that "simple" steps in the space of sequences (like changing the pitch of a quarter note from a C4 to a D4) require many iterations of Gibbs sampling procedure if musical data is encoded using a piano roll representation while it takes less iterations using the melodico-rhythmic encoding.

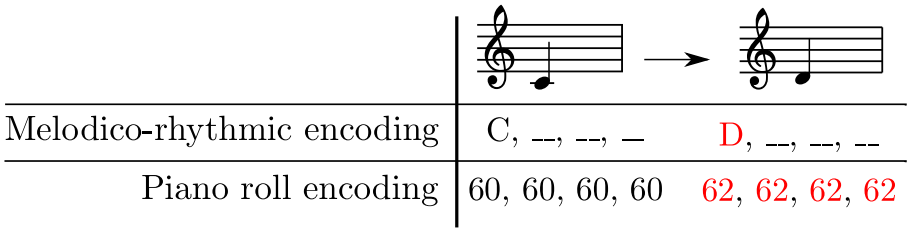


Figure 27: Comparison between the melodico-rhythmic encoding and the piano roll encoding in order to change a C quarter note into a D quarter note. Less steps are required to make simple steps in the space of sequences. This makes the melodico-rhythmic encoding compatible with the proposed pseudo-Gibbs sampling.

Implementation Details

We implemented DeepBach using Keras [29] with the Tensorflow [1] backend. We used the database of chorale harmonizations by J.S. Bach included in the music21 toolkit [38]. After removing chorales with instrumental parts and chorales containing parts with two simultaneous notes (bass parts sometimes divide for the last chord), we ended up with 352 pieces. Contrary to other approaches which transpose all chorales to the same key (usually in C major or A minor), we choose to augment our dataset by adding all chorale transpositions which fit within the vocal ranges defined by the initial corpus. This gives us a corpus of 2503 chorales and split it between a training set (80%) and a validation set (20%). The vocal ranges contains less than 30 different pitches for each voice (21, 21, 21, 28) for the soprano, alto, tenor and bass parts respectively.

As shown in Fig. 26, we model only *local* interactions between a note v_i^t and its context $(v_{\setminus i,t}, \mathcal{M})$ i.e. only elements with time index t between $t - \Delta t$ and $t + \Delta t$ are taken as inputs of our model for some

scope Δt. This approximation appears to be accurate since musical analysis reveals that Bach chorales do not exhibit clear long-term dependencies.

The reported results in Sect. 6.3 and examples in Sect. 6.4.3 were obtained with $\Delta t = 16$. We chose as the "neural network brick" in Fig. 26 a neural network with one hidden layer of size 200 and ReLU [110] non-linearity and as the "Deep RNN brick" two stacked LSTMs [69, 105], each one being of size 200 (see Fig. 2 (f) in [92]). The "embedding brick" applies the same neural network to each time slice $(\mathcal{V}_t, \mathcal{M}_t)$. There are 20% dropout on input and 50% dropout after each layer.

We experimentally found that sharing weights between the left and right embedding layers improved neither validation accuracy nor the musical quality of our generated chorales.

EXPERIMENTAL RESULTS

We evaluated the quality of our model with an online test conducted on human listeners. These experiments were made with an early version of the DeepBach algorithm where notes were using their MIDI pitches.

Setup

For the parameters used in our experiment, see Sect 6.2.4. We compared our model with two other models: a Maximum Entropy model (MaxEnt) as in [63] and a Multilayer Perceptron (MLP) model.

The Maximum Entropy model is a neural network with no hidden layer. It is given by:

$$p_i(\mathcal{V}_i^t | \mathcal{V}_{\setminus i,t}, \mathcal{M}, A_i, b_i) = \text{Softmax}(AX + b) \tag{29}$$

where X is a vector containing the elements in $\mathcal{V}_{\setminus i,t} \cup \mathcal{M}_t$, A_i a (n_i, m_i) matrix and b_i a vector of size m_i with m_i being the size of X, n_i the number of notes in the voice range i and Softmax the softmax function given by

$$\text{Softmax}(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j \in [K],$$

for a vector $z = (z_1, \ldots, z_K)$.

The MLP model that we chose takes as input elements in $\mathcal{V}_{\setminus i,t} \cup \mathcal{M}$, is a neural network with one hidden layer of size 500 and uses a ReLU [110] non-linearity.

All models are local and have the same scope Δt, see Sect. 6.2.4.

Subjects were asked to give information about their musical expertise. They could choose what category fits them best between:

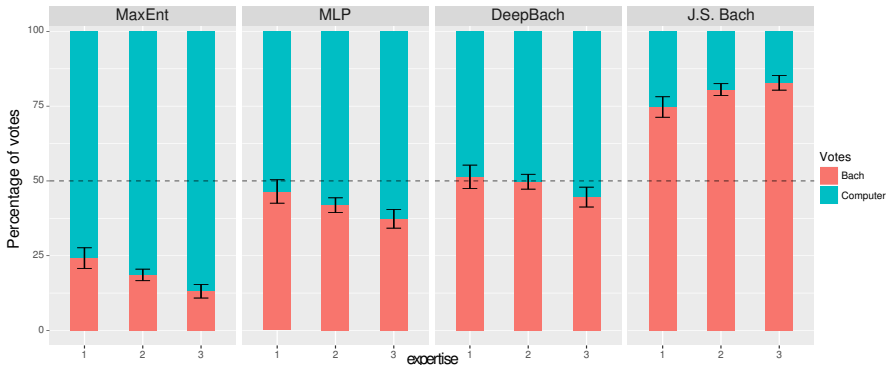


Figure 28: Results of the "Bach or Computer" experiment. The figure shows the distribution of the votes between "Computer" (blue bars) and "Bach" (red bars) for each model and each level of expertise of the voters (from 1 to 3), see Sect. 6.3.2 for details.

1. I seldom listen to classical music
2. Music lover or musician
3. Student in music composition or professional musician.

The musical extracts have been obtained by reharmonizing 50 chorales from the validation set by each of the three models (MaxEnt, MLP, DeepBach). We rendered the MIDI files using the Leeds Town Hall Organ soundfont[4] and cut two extracts of 12 seconds from each chorale, which gives us 400 musical extracts for our test: 4 versions for each of the 100 melody chunks. We chose our rendering so that the generated parts (alto, tenor and bass) can be distinctly heard and differentiated from the soprano part (which is fixed and identical for all models): in our mix, dissonances are easily heard, the velocity is the same for all notes as in a real organ performance and the sound does not decay, which is important when evaluating the reharmonization of long notes.

Discrimination Test: the "Bach or Computer" experiment

Subjects were presented series of only one musical extract together with the binary choice "Bach" or "Computer". Fig. 28 shows how the votes are distributed depending on the level of musical expertise of the subjects for each model. For this experiment, 1272 people took this test, 261 with musical expertise 1, 646 with musical expertise 2 and 365 with musical expertise 3.

The results are quite clear: the percentage of "Bach" votes augment as the model's complexity increase. Furthermore, the distinction between computer-generated extracts and Bach's extracts is more accu-

4 https://www.samplephonics.com/products/free/sampler-instruments/the-leeds-town-hall-organ

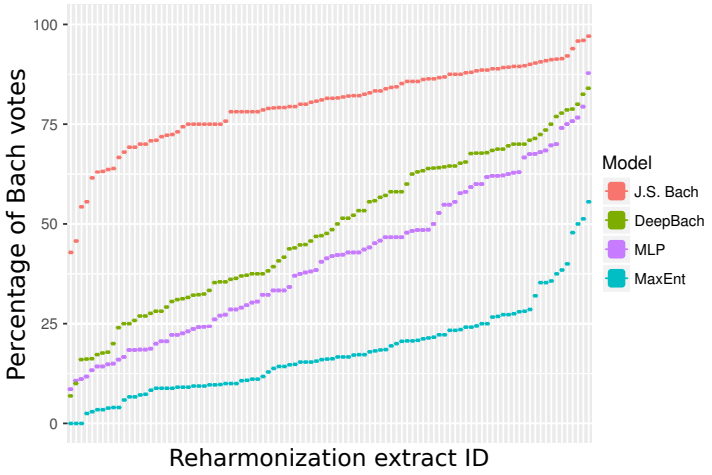


Figure 29: Results of the "Bach or Computer" experiment. The figure shows the percentage of votes for Bach for each of the 100 extracts for each model. For each model, a specific order for the x-axis is chosen so that the percentage of Bach votes is an increasing function of the x variable, see Sect. 6.3.2 for details.

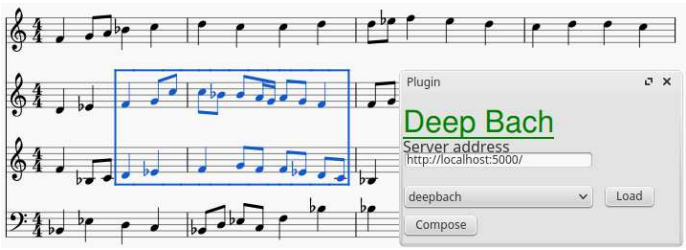


Figure 30: DeepBach's plugin minimal interface for the MuseScore music editor

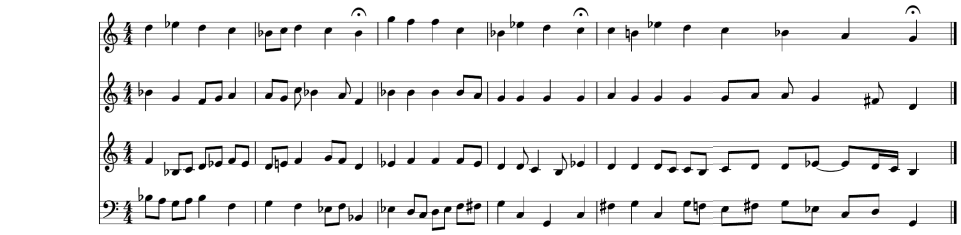
rate when the level of musical expertise is higher. When presented a DeepBach-generated extract, around 50% of the voters would judge it as composed by Bach. We consider this to be a good score knowing the complexity of Bach's compositions and the facility to detect badly-sounding chords even for non musicians.

We also plotted specific results for each of the 400 extracts. Fig. 29 shows for each reharmonization extract the percentage of Bach votes it collected: more than half of the DeepBach's automatically-composed extracts has a majority of votes considering them as being composed by J.S. Bach while it is only a third for the MLP model.

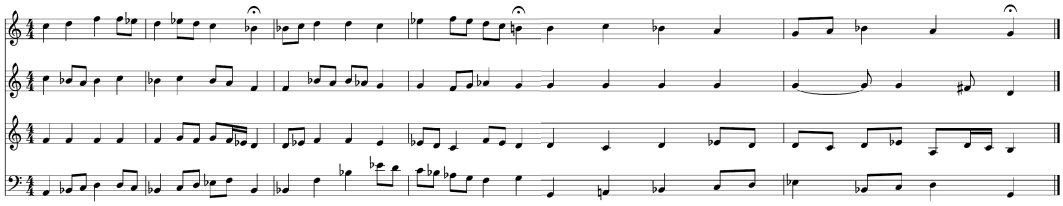
INTERACTIVE COMPOSITION

Description

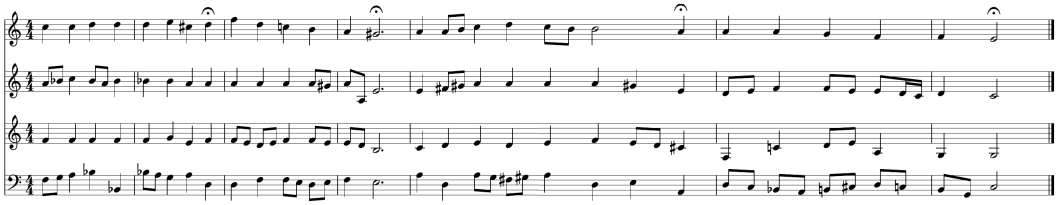
We developed a plugin on top of the MuseScore music editor allowing a user to call DeepBach on any rectangular region of the score.



(a)



(b)



(c)

Figure 31: Examples produced using DeepBach as an interactive composition tool. Examples (a) and (b) share the same metadata.

Even if the interface is minimal (see Fig. 30), the possibilities are numerous: we can generate a chorale from scratch, reharmonize a melody and regenerate a given chord, bar or part. We believe that this interplay between a user and the system can boost creativity and can interest a wide range of audience.

Adapting the model

We made two major changes between the model we described for the online test and the interactive composition tool.

Note encoding

We changed the MIDI encoding of the notes to an encoding of the notes using their *full name*. As explained in Sect. 3.1.1, some information is lost when reducing a music sheet to its MIDI representation since we cannot differentiate between two *enharmonic notes*. We noticed that this improvement enables the model to generate notes with the correct spelling and prevented the model to make some unexpected modulations.

Steering modulations

We added the *current key signature* list $\mathcal{K}$ to the metadata $\mathcal{M}$. This allows users to impose modulations and key changes. Each element $\mathcal{K}_t$ of this list contains the number of sharps of the estimated key for the current bar. It is a integer between -7 and 7. The current key is computed using the key analyzer algorithm from music21.

Generation examples

We now provide and comment on examples of chorales generated using the DeepBach plugin. Our aim is to show the quality of the solutions produced by DeepBach. For these examples, no note was set by hand and we asked DeepBach to generate regions longer than one bar and covering all four voices.

Despite some compositional errors like parallel octaves, the musical analysis reveals that the DeepBach compositions reproduce typical Bach-like patterns, from characteristic cadences to the expressive use of nonchord tones. As discussed in Sect. 6.4.2, DeepBach also learned the correct spelling of the notes. Among examples in Fig. 31, examples (a) and (b) share the same metadata ($\mathcal{S}, \mathcal{F}$ and $\mathcal{K}$). This demonstrates that even with fixed metadata it is possible to generate contrasting chorales.

Since we aimed at producing music that could not be distinguished from actual Bach compositions, we had all provided extracts sung by the Wishful Singing choir. These audio files can be heard on the accompanying website[5].

DISCUSSION AND FUTURE WORK

We described DeepBach, a probabilistic model together with a sampling method which is flexible, efficient and provides musically convincing results even to the ears of professionals. The strength of our method is the possibility to let users impose unary constraints, which is a feature often neglected in probabilistic models of music. Through our graphical interface, the composition of polyphonic music becomes accessible to non-specialists. The playful interaction between the user and this system can boost creativity and help explore new ideas quickly. We believe that this approach could form a starting point for a novel compositional process that could be described as a constructive dialogue between a human operator and the computer. This method is general and its implementation simple. It is not only applicable to Bach chorales but embraces a wider range of polyphonic music.

5 https://sites.google.com/site/deepbachexamples/

Future work aims at refining our interface, speeding up generation and handling datasets with small corpora.

Part III

NOVEL TECHNIQUES IN SEQUENCE GENERATION

7

DEEP RANK-BASED TRANSPOSITION-INVARIANT DISTANCES ON MUSICAL SEQUENCES

INTRODUCTION

Determining whether two musical sequences are similar or not is a key ingredient in music composition. Indeed, the repeated occurrences of a given pattern (transformed or not) is easily perceived by an attentive listener. Among possible transformations of a pattern, we can cite

- transposition
- mode change
- augmentation / diminution
- ornamentation / simplification.

These patterns in music gives the listener expectations of what could follow. This latter is then gratified to have guessed right or can be surprised by a pleasing or unexpected change. The musical pieces may then appear more coherent and organized. It is then up to the composer to play with this series of fulfilled/unfilled expectations. From a compositional point of view, this allows a composer to create long pieces of music while retaining the listener's attention.

Many musical pieces are intrinsically-based on the different repetitions of a given pattern. Fugues or sonatas are examples of such pieces where the overall structure results from how the occurrences and transformations of a given pattern unfold through time. But this is also true on a local scale in many musical genres since patterns give coherence between musical phrases. This is particularly observable in pop and jazz songs (see Fig. 2 for instance for an example of a pattern and its transformation).

Due to the omnipresence of patterns in music, good distances on musical sequences are essential and can be used for a wide variety of tasks:

- plagiarism detection [23]
- music retrieval [3]
- automatic musical analysis [40, 56]
- automatic music generation [132].

The traditional distance used for sequence comparison is the edit-distance (also known as the Levenshtein distance [91]). On sequences of symbols, it basically consists in assigning a cost to different basic edit operations such as the insertion, the deletion or the substitution of a single symbol. The distance between two sequences is then the cost of the minimal sequences of basic edit operations allowing to change the first sequence into the second one. This minimal cost is easily calculated using a dynamic programming algorithm.

Most of the existing distances on musical sequences are based on generalizations of the aforementioned distance by taking into account the specificity of the set of musical sequences (the edit distance was primarily designed for text and widely used in biology).

For instance, [107] proposes to extend the set of basic edit operations with two other operations that are more specific to musical sequences: fragmentation and consolidation.

The main issue with this approach is that the edit-distance strongly depends on the choice of the encoding of the musical content. Indeed, contrary to textual data, musical content (such as monophonic melodies) can be encoded in *many ways* and there are *a priori* no representation which is better than another.

The importance of the choice of the data representation is pinpointed in [21]. In this paper, the authors argue that the MIDI pitch representation is insufficient for applications in tonal music as it disregards the hierarchical importance of diatonic scale tones over the 12-tone discrete pitch space. To address this issue, they propose a new representation, called the General Pitch Interval Representation. It is a representation that takes into account the diatonic intervals in scale steps and other more abstract representations such as contour strings (the contour string of a melody is a representation where only the following events are considered: repeat, ascending or descending step, ascending or descending leap).

The interest in such a representation is that it introduces perceptually salient information directly into the sequence encoding.

This idea is further explored in [59] where they propose to encode a melody using its Implication/Realization structure. It is a concept drawn from the theory of perception and cognition of melodies from [111] which is based on the Gestalt theory. It consists in assigning different basic structures depending on the contour of a sequence and can be considered as a generalization of the contour string encoding.

Using contour strings encoding imply that a melody is only considered up to a transposition since only intervals with respect to the previous note is considered [90]. The same idea can also be used by considering ratios of rhythmic patterns instead of their absolute values. We refer the reader to [64] where the authors study the advantages and drawbacks of many monophonic sequence representations on the edit-distance algorithms.

Geometric interpretations of the distance between two musical sequences have also been proposed [102, 151]. Their advantage is to be applicable on polyphonic sequences contrary to the methods based on the edit-distance. [89] compares the two approaches on a variety of music retrieval tasks.

However, defining how close two sequences are is in fact an ill-defined problem. This notion is very *subjective* and it seems implausible to find a universal rule applying to every musical style and sequences: sequences can be "close" with respect to a given music style and "far" in another music style. Furthermore, attempts to ground it on a psychological basis using an appropriate representation still suffer from the arbitrariness of the underlying distance.

In this work, we introduce a corpus-dependent distance between two musical sequences. Our distance relies on a permutation-based distance [7] applied on high-level features obtained via a sequence-to-sequence autoencoder. This approach is general and we believe more independent of the choice of the representation than all previous methods, and can be applied to both monophonic and polyphonic music.

We then extend our method in order to obtain a way to generate transposition-invariant distances, which means that a sequence and its transposition should be close. Contrary to other methods, this distance is made transposition-invariant without changing the sequence encoding.

In the following, we focus on monophonic sequences with a given representation, but these ideas can easily be generalized to other representations of musical material, from monophonic to polyphonic ones.

Our contributions are the following:

- introduction of a framework to build distances on sequences,
- introduction of corpus-dependent musical distances in music,
- extension of this approach in order to construct invariant distances with respect to a given set of transformations.

We believe that linking the distance between musical sequences to the specific genre of these musical sequences is a way to address issues related to the choice of a perceptually-appropriate distance and is more likely to yield better results.

The outline of this paper is the following: In Sect. 7.2, we expose related works about transformation-invariant features and transformation-invariant distances; Section 7.3 introduces our method to construct a data-dependent distance on sequences and Sect. 7.4 improves this method in order to obtain a distance which is invariant with respect to a set of transformations; Finally, in Sect. 7.5 we present experimental results about the introduced distances and highlight their

efficiency on the dataset of the *chorale melodies from the J.S. Bach chorale harmonizations.*

RELATED WORKS

Finding transformation-invariant distances is a problem which was primarily addressed on image datasets. Indeed, learning distances from a corpus of images is crucial in many applications (classification, clustering, or retrieval tasks) and it is often desirable that this distance be independent under some natural transformations on images such as rotations, scalings and translations.

Taking into account the "natural" set of transformations which acts on a dataset is of a great interest since we can use this information to obtain more robust and more informative feature representations.

The feature representations need not necessarily be invariant with respect to a set of transformations, but sometimes only equivariant [34]. Equivariance here means that the feature representation of a transformed image can be obtained by applying a known transformation directly to the feature representation of the original image; that is

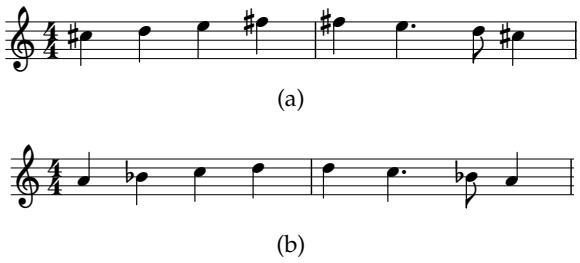
$$\phi(t.x) = t.\phi(x), \quad \forall t \in T \tag{30}$$

for a feature map $\phi : X \to Y$ and a group of transformation T acting on X and Y.

Convolutional Neural Networks (CNNs) [82, 140] for instance take into account the importance of translations on image datasets by using the same transposed filter. A generalization of the regular CNN filters has been proposed [161]. It aims at obtaining a CNN which is equivariant to both translations and rotations. It is in fact possible to devise more general approaches which are able to deal with any symmetry group acting on images as shown in [16, 33, 53]. In [16], the proposed model is suitable for a theoretical analysis about the stability of the learned invariant representations.

However, equivariant feature representations are not particularly suitable for building invariant feature representations. Indeed, in order to obtain an invariant representation, one would have to average all feature representations of all possible image transformations under the chosen group, which is either intractable or computationally demanding.

An approach in the context of shape matching which shares the same motivations as ours can be found in [99]. Contrary to images, two shapes are considered to be identical if we can obtain one by applying a group transformation on the other one. These group transformations can be as above the group of displacements (translations and rotations), but can also be, in the context of shapes, the affine group (translations, rotations and rescalings). A distance between shapes is



(a)

(b)

Figure 32: (a) a 2-bar sequence and (b) its transposition a major third lower.

then constructed by introducing a distance on the integral invariants of a shape. These handcrafted quantities are invariant with respect to the group of transformations acting on shape contours and thus assert that the constructed distance is well-defined on shapes. The main difference with the approaches in image is that the feature representations (here the integral invariants) is not learned from data but constructed by hand.

In comparison, our method is able to learn transformation-invariant feature representations from data resulting in a transformation-invariant distance.

CORPUS-BASED DISTANCE

Rank-based distance

Our method to construct distances on sequences relies on the rank-based (or permutation-based) distances described in [7]. The idea is to define a distance solely based on the ranking of the high-level activations of a Deep Neural Network (DNN) [58]. Indeed, activations of neurons from high-level hidden layers can be used as features for describing the input. Each of this feature can encode a particular concept about the input and how these concepts are ranked with respect to one another is sufficient to determine if two inputs are similar.

More precisely, let x be a N-dimensional feature vector. We define $\Pi(x)$ to be the vector of ranks of x. This vector is a permutation of the N-tuple $[N]$ such that

$$\forall i, j \in [N], \quad i < j \implies x_{\Pi(x)_i} \geqslant x_{\Pi(x)_j}. \tag{31}$$

With this notation, $\Pi(x)_1$ is the index i such that x_i is the greatest coordinate of x and $\Pi(x)_N$ the index of the smallest coordinate of x.

Given two feature vectors x and y, we can define their distance to be the distance between their permutations $\Pi(x)$ and $\Pi(y)$ using popular distances between permutations. In the following, we will consider two popular rank correlation measures: Spearman's rho distance and Kendall's tau distance.

In its simplest formulation, Spearman's rho distance ρ between feature vectors x and y is defined to be the ℓ_2 norm between their vectors of ranks. This gives us:

$$\rho(x, y) = \sqrt{\sum_{i=1}^{N} (\Pi(x)_i - \Pi(y)_i)^2}. \tag{32}$$

The Kendall tau distance is a measure of similarity between two rank variables. It is based on the number of pairwise inversions needed to change one ordering into the other. Given two vectors of ranks $\Pi(x), \Pi(y)$ of size N, we say that a pair of integers $i < j$ is

- concordant if $\Pi(x)_i < \Pi(x)_j$ and $\Pi(y)_i < \Pi(y)_j$,
- discordant if $\Pi(x)_i < \Pi(x)_j$ and $\Pi(y)_i > \Pi(y)_j$,

and these definitions also hold when we reverse all inequalities. Since there are $N(N-1)/2$ such pairs, we can define a similarity on feature vectors x and y by

$$\tau(x, y) = \frac{\sharp\{\text{concordant pairs}\} - \sharp\{\text{discordant pairs}\}}{N(N-1)/2}, \tag{33}$$

where $\sharp\{\text{concordant pairs}\}$ indicates the number of concordant pairs when considering the rank vectors of x and y. This similarity measure is in $[-1, 1]$ and equals 1 when the ranks are all equal.

Sequence-to-Sequence autoencoder

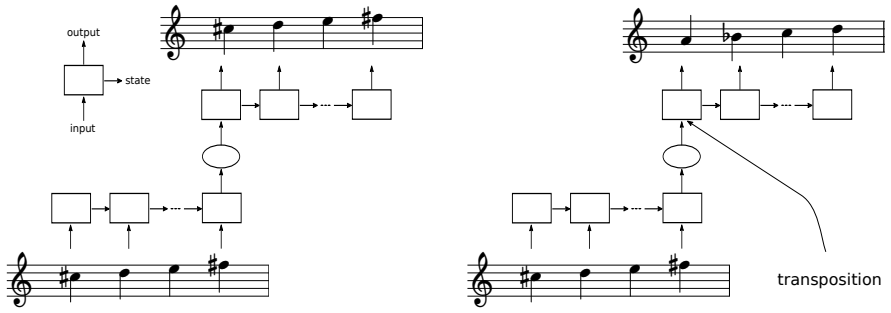
In the following, we consider that we have a i.i.d. dataset $\mathcal{D}$ of K sequences of length L

$$\mathcal{D} := \left\{ s^{(k)} = (s_1^{(k)}, \dots, s_L^{(k)}), \quad s_i \in [A] \right\}_{k \in [K]}, \tag{34}$$

where sequences are composed of tokens in $[A]$ with A the size of the alphabet. The objective is to obtain a mapping from the space of sequences A^L to a feature representation in $\mathbf{R}^N$.

In order to build a high-level representation of musical sequences in an unsupervised manner, we consider using a *sequence-to-sequence* model [28, 146] (Fig. 33a) as an autoencoder [58]. An autoencoder is a Neural Network (NN) parametrized by θ which is composed of two parts: an encoding NN enc_θ which usually maps the high-dimensional observation space to a feature representation (or code) of smaller dimensionality and a decoding NN dec_θ whose aim is to predict the original sequence given its code.

In our case, these neural networks are implemented using Recurrent Neural Network (RNN) [58]. The feature representation $\text{enc}_\theta(s)$ for a sequence s is obtained by considering only the output of the RNN on the last time step.



(a) sequence-to-sequence autoencoder (b) transposing sequence-to-sequence architecture

Figure 33: Sequence-to-sequence autoencoder and its generalization. Boxes represent RNN cells. Only the output of the last RNN cell is used. The feature representation is displayed as an oval.

The decoder returns a probability distribution $\pi = (\pi_1, \dots, \pi_L)$ over the sequences in A^L where each $\pi_i = (\pi_{i,1}, \dots, \pi_{i,A})$ is a categorical distribution over $[A]$ ($\pi_{i,a} \geqslant 0$ and $\sum_{a=1}^{A} \pi_{i,a} = 1$ for all i).

The parameters θ of the RNN are chosen to minimize the following loss:

$$\mathcal{L}(\theta; \mathcal{D}) := \sum_{k=1}^{K} \sum_{i=1}^{N} H\left[s_i^k; \mathrm{dec}_\theta(\mathrm{enc}_\theta(s^k))_i\right] \tag{35}$$

where

$$H\left[s_i^k; \mathrm{dec}_\theta(\mathrm{enc}_\theta(s^k))_i\right] := - \sum_{a=1}^{A} \delta_a(s_i^k) \log\left[\mathrm{dec}_\theta(\mathrm{enc}_\theta(s^k))_{i,a}\right], \tag{36}$$

denotes the categorical cross-entropy with δ_a such that $\delta_a(x) = 1$ iff. $x = a$ and 0 otherwise.

Details about our implementation are discussed in Sect. 7.5.1.

ReLU non-linearity and truncated Spearman rho distance

In exposing the rationale behind the permutation-based distance in Sect. 7.3.1, we put forward the idea that each feature of a sequence encodes a particular high-level concept. An ordering of these concepts would then act like a fingerprint for this sequence. However, in the model described above, the "most-relevant concepts" (the coordinates x_i which have supposedly the greatest impact on the sequence decoding) are the coordinates with the greatest absolute value. Coordinates near zero are unlikely to be particularly relevant but have a great influence on the ordering $\Pi(x)$ of Eq. 31 and on the distance Eq. 32.

A way to deal with this problem consists in adding a Rectifier Linear Unit (ReLU) activation [110] on top of the encoder RNN. The resulting feature vector will then contain only non-negative elements with a substantial number of null elements. In doing so, we impose that the "most-relevant concepts" are within the first values of the permutation $\Pi(x)$. It is then possible to modify the Spearman rho distance by only considering the l most important coordinates, which results in the truncated Spearman rho distance of order $l < N$:

$$\rho_l(x, y) = \sqrt{\sum_{i=1}^{l} (\Pi(x)_i - \Pi(y)_i)^2}. \tag{37}$$

We finally define the corpus-dependent distance $D_{\mathcal{D}}$ between two sequences s and s' truncated up to l by

$$D_{\mathcal{D}}(s, s'; l) = \rho_l(\text{enc}_\theta(s), \text{enc}_\theta(s')), \tag{38}$$

where enc_θ is the encoder RNN (with a ReLU non-linearity on its top-most layer) of a trained sequence-to-sequence autoencoder.

TRANSFORMATION-INVARIANT DISTANCES

We now suppose that we have a set of transformations $\mathcal{T}$ that act on sequences. We suppose that this action defines an *equivalence relation* on $\mathcal{D}$. This means that if there exists $t \in \mathcal{T}$ such that $s = t.s'$ for sequences $s, s' \in \mathcal{D}$, then there exists $t' \in \mathcal{T}$ such that $s' = t'.s$. In such a case, we note $s \sim s'$ and their *equivalence class* is notated $\mathcal{T}.s$.

A typical example on musical sequences would be the set of transpositions (see Fig. 36). The sequence dataset is then split between the different equivalence classes $\mathcal{T}.s$. Our objective is to devise a distance between sequences that would be invariant relatively to this set of transformations, i.e. the distance only depends on the equivalence classes and not on the sequences themselves.

A simple way to achieve this goal is to directly obtain transformation-invariant feature representations. For this, we need to modify the preceding architecture so that the feature representation (represented as a circle in Fig. 33a) is the same for a sequence and all its transformations. We introduce the sequence-to-sequence architecture depicted in Fig. 33b. It is a model which takes as an input a sequence $s \in \mathcal{D}$ and a transformation $t \in \mathcal{T}$ and learns to predict the transformed sequence $t.s$. In this model, the encoder $\text{enc}_\theta(s)$ is only a function of s while the decoder $\text{dec}_\theta(x, t)$ takes as an input a feature representation x together with the transformation t applied to s.

This architecture is trained using the following loss function:

$$\mathcal{L}(\theta; \mathcal{D}, \mathcal{T}) := \sum_{t \in \mathcal{T}_k} \sum_{k=1}^{K} \sum_{i=1}^{N} H\left[(t.s^k)_i; \mathrm{dec}_\theta(\mathrm{enc}_\theta(s^k), t)_i\right], \quad (39)$$

where the first sum over $t \in \mathcal{T}_k$ denotes the set of transformations $t \in \mathcal{T}$ such that $t.s^k \in \mathcal{D}$.

Note that, in the case of musical transpositions, we can specify how we want to transpose our input sequence s by two means:

- *relatively to* s, by specifying the interval by which we transpose s (relative representation),
- *independently of* s, by specifying, for instance, the name of the starting note of $t.s$ (absolute representation).

Since we want a feature representation which depends only on equivalence classes $T.s$ and not on its representatives $s' \in T.s$, we must use an absolute representation when specifying the transformations $t \in \mathcal{T}$. Indeed, using a relative representation would otherwise force the feature representation to contain absolute information. Plugging this representation in a rank-based distance would lead to a distance which is not transformation-invariant.

Even when doing so, the distances we obtain are not fully-invariant: the feature representation can still contain absolute information about sequence s. We propose to overcome this issue by forcing the model to compute averaged feature representations. Ideally, using the mean representation

$$\overline{\mathrm{enc}}_\theta(\mathcal{T}.s) := \frac{1}{|\mathcal{T}.s|} \sum_{s' \in \mathcal{T}.s} \mathrm{enc}_\theta(s') \quad (40)$$

for an equivalence class $\mathcal{T}.s$ gives a transformation-invariant representation, but it is computationally-expensive. A simple approximation is to compute this mean representation using only two sequences belonging to the same equivalent class. For two sequences $s, s' \in \mathcal{T}.s$, we consider the averaged representation

$$\overline{\mathrm{enc}}_\theta(s, s') := \frac{1}{2}\left(\mathrm{enc}_\theta(s) + \mathrm{enc}_\theta(s')\right); \quad (41)$$

this representation is then passed to the decoder together with the absolute transformation representation. This architecture is represented in Fig. 34.

We finally add a ℓ_1-penalty on the difference between the two feature representations to encourage the model to make these two representations equal. The loss function we obtain in this case is then given by

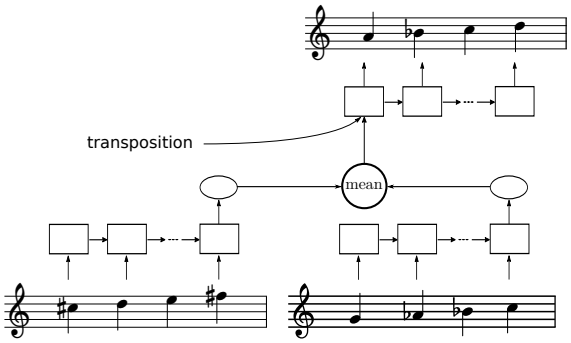


Figure 34: Proposed architecture for computing transformation-invariant feature representations. The input and output sequences belong to the same equivalence class.

$$\mathcal{L}_{\text{invariant}}(\theta; \mathcal{D}, \mathcal{T}, \lambda) :=$$
$$\sum_{t,t' \in \mathcal{T}_k} \sum_{k=1}^{K} \sum_{i=1}^{N} \left(H\left[(t'.s^k)_i \, ; \text{dec}_\theta\left(\overline{\text{enc}}_\theta\left(s^k, t.s^k\right), t'\right)_i \right] + \right.$$
$$\left. \lambda \ell_1 \left[\text{enc}_\theta(s^k) - \text{enc}_\theta(t.s^k) \right] \right), \quad (42)$$

where λ is a hyper-parameter controlling the trade-off between accuracy and enforcing the invariance property.

We thus define a transformation-invariant $D_{\mathcal{D},\mathcal{T}}$ distance relative to a corpus $\mathcal{D}$ and a set of transformations $\mathcal{T}$ as in Eq. 38, except that the encoder network enc_θ is the encoder network of the modified architecture schematically displayed in Fig. 34 and trained with loss $\mathcal{L}_{\text{invariant}}(\theta; \mathcal{D}, \mathcal{T}, \lambda)$ given by Eq. 42.

This transposition-invariant distance has the advantage that the notion of invariance is directly encoded into the learned model. Indeed, a simpler way to produce a transposition-invariant distance would be to apply the corpus-based distance of Eq. 38 to a corpus where all sequences would start with the same note, say C4. The distance between any two sequences would thus be the distance between their transposed version starting in C4. However, these transpositions cannot be implemented effectively: the first note has to be found in order to know how to transpose (we can have rests or hold symbols, see Sect. 7.5.1). This operation takes some time and it cannot be easily parallelizable on a GPU. The proposed transposition-invariant distance thus transfer these costly operations from the evaluation phase to the data-preprocessing phase.

EXPERIMENTAL RESULTS

We report experiments on the *chorale melodies* from the *chorale harmonizations by J.S. Bach* presented in Sect. 2.3.4. This dataset is obtained

by extracting all soprano parts from the J.S. Bach chorales dataset included in the music21 [38] Python package.

Implementation details

We choose to encode our data using the melodico-rhythmic encoding described in Sect. 3.1.3. More specifically, time is quantized with sixteenth notes and we use the *full name* of the notes instead of the traditional MIDI pitch representation. We introduce two additional tokens in order to handle rhythm and pitch in a unified fashion: a hold symbol HOLD indicating that a note is being played but not attacked, and a rest symbol REST. In this setting, a musical sequence is only an ordered sequence of tokens drawn from the set of all possible notes {C3, C#3, Db3, D3, D#3, ...} together with the HOLD and REST tokens.

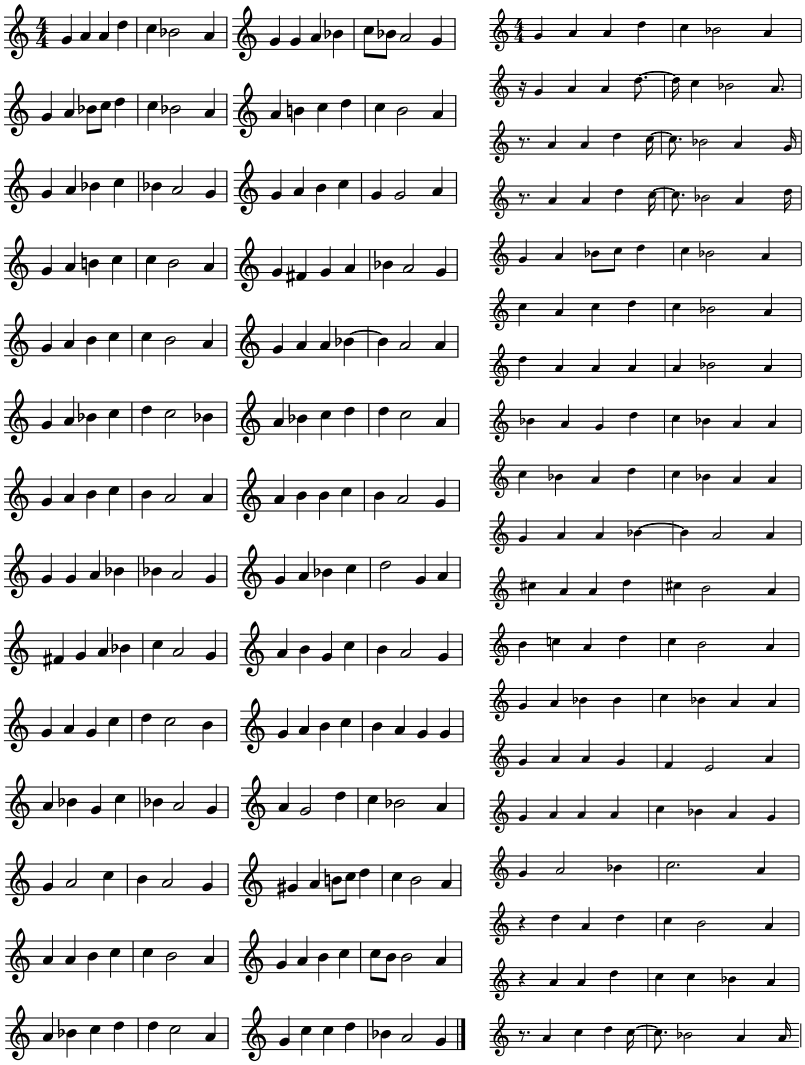
The set of transformations $\mathcal{T}$ we choose is the set of all transpositions. But for a given sequence s, we only define as its equivalence class the set of its transpositions which fits within the original voice range. In doing so, we do not need the set of transformations to be a group, but require only that it defines an equivalence relation.

The RNN we use is a 2-layer stacked LSTM [69] with 512 hidden units each. The ReLU non-linearity is used and the truncation order l of Eq. 37 is set to 256.

Nearest neighbor search

We empirically demonstrate the efficiency of these distances by displaying examples of nearest neighbor requests. Fig. 35a shows examples of melodies which are "close" according to the corpus-dependent distance $D_{\mathcal{D}}$. All the results that we display are obtained using Spearman's rho distance, but we obtain similar results by replacing it with Kendall's tau similarity measure.

The nearest neighbors query of Fig. 35a reveals interesting behaviors of the corpus-dependent distance $D_{\mathcal{D}}$. The overall shape of the target melody (ascending then descending) is present in every neighboring sequence under various aspects. There are interesting rhythmic variations and also key changes. But, what is the most striking here, is that some other characteristic elements of the target sequence are also taken into account. For instance, the importance of the ascending leap on the fourth beat is present in many sequences. Another such example is the note repetition at beats 2 and 3 which also occurs in some neighboring sequences. We believe that the last sequence in the example displayed in Fig. 35a is a good illustration of how characteristic elements are captured with our distance. Even if there are only two notes in common with the target sequence, we still



(a) Corpus-dependent distance $D_{\mathcal{D}}$ constructed using Spearman's rho distance

(b) Edit distance

Figure 35: A target melody and its nearest neighbors according to different distances on sequences. Duplicates have been removed.

have an impression of proximity between both. This may be due to the following facts

- the most important notes of the target sequence (the highest and the lowest) are replicated,
- the key and the rhythm are identical,
- there is an ascending fourth in both (on beat 2 for the neighboring sequence instead of on beat 4 for the target sequence),
- there is a note repetition on beats 2 and 3,
- they both conclude by a descending conjunct movement.

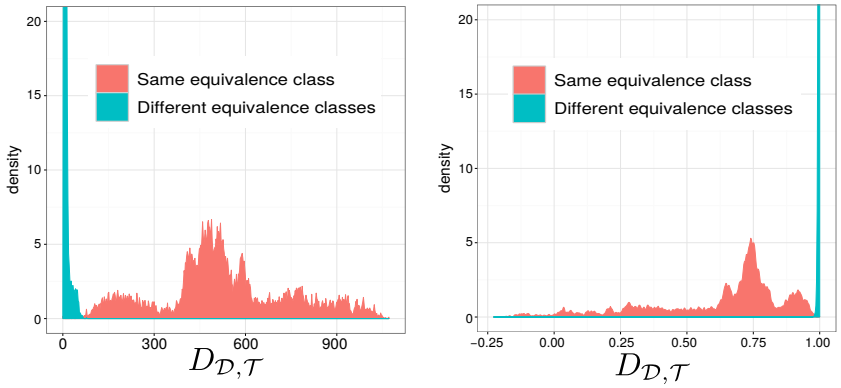
This has to be compared with the nearest neighbors returned using the edit distance on the same target sequence. This is displayed in Fig. 35b. This presents some unwanted behaviors. For instance, sequences identical to the target sequence but with a sixteenth note offset are considered to be almost identical. However, from a musical point of view, the importance of the difference between notes on and notes off the beat is crucial. In the other cases, the edit only manages to find sequences containing common notes (and played at the exact same time) with the target sequence. Since the HOLD symbol is seen as note like any other one, we can see that replacing a half note by a quarter note has a cost of only one, independently of the chosen note. The importance here is not on the melodic contours nor on an intuitive perception of similarity.

The behavior of the edit distance seems more erratic and less predictable. It is also "less smooth" than our proposed distances (see Fig. 36) since the edit distance can only take a finite (and smaller) number of values. The important difference is that the number of possible values returned by the edit distance depends on the sequences size, while it depends on the feature vector size in the case of corpus-dependent distances.

Invariance by transposition

In this section, we check to which extent the distance $D_{\mathcal{D},\mathcal{T}}$ is invariant under the set of transformations $\mathcal{T}$. In Fig. 36, we plot the distance $D_{\mathcal{D},\mathcal{T}}(s, s')$ when s and s' are in the same equivalence class under $\mathcal{T}$ and when they are not.

Two things are two be noted in this example. Firstly, the obvious difference between the two densities and the sharp peak when the sequences are in the same equivalence class show that our distance indeed captures the invariance by transposition on musical sequences. Secondly, the behavior of the density of the distance between two randomly-chosen sequences is interesting: it is multimodal and widespread. This distance can take numerous different values and



(a) Using Spearman's rho rank distance (b) Using Kendall's tau rank distance

Figure 36: Density estimations of distances $D_{\mathcal{D},\mathcal{T}}(s, s')$ between two sequences s and s' belonging to the same or to different equivalence classes. The two proposed rank distances are displayed. Our architecture allows to capture the invariance by transposition for musical sequences (peaked blue). The images are truncated for clarity.

can then have more discriminative power. We have experimentally seen in Sect. 7.5.2 that the corpus-dependent distance is able to capture high-level musical concepts. It is the same for its transposition-invariant counterpart.

To show this, we do not replicate the results about the nearest neighbor search of Fig. 35a since it only returns exact transpositions of the target sequence. Instead, we only make a nearest neighbor search on a small subset of the J.S. Bach chorales using $D_{\mathcal{D},\mathcal{T}}$. The result is displayed in Fig. 37.

The same analysis as the one conducted for Fig. 35a can be made, except that it is now irrespective of the transposition of the neighboring sequences. We conclude that this transposition-invariant distance allows to detect characteristic patterns and musical motives independently of the key they are in.

CONCLUSION

We proposed a novel framework to build distances learned from musical corpora. Because they take into account the style to which musical sequences belong, these learned distances are not subject to the usual problems encountered using the edit-distance generalizations: they are less dependent on the input encoding while being more satisfactory from a perceptive point of view. Indeed, using neural-network-learned features instead of handcrafted features allows to define a natural notion of proximity between sequences which is rooted on an objective and non-biased a priori basis. The choice of the rank-based distance applied on the feature representations does not seem

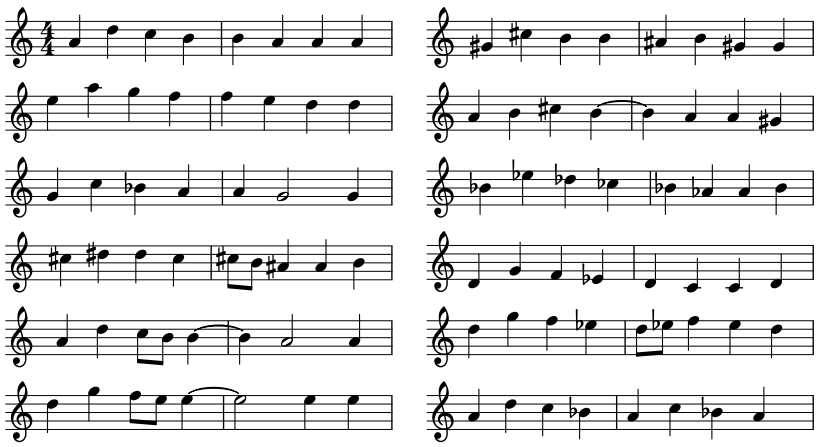


Figure 37: A target melody and its nearest neighbors among 20000 sequences randomly drawn from all possible subsequences using the transposition-invariant distance $D_{\mathcal{D},\mathcal{T}}$ based on the Spearman rank-based distance.

to influence much the final distance behavior. This framework can be modified so that the distances we obtain are invariant with respect to a given a set of transformations. This is made possible by the fruitful combination between a quasi-invariant feature representation learned from a regularized sequence-to-sequence architecture and a rank-based distance. Since the feature representations are not necessarily equal for sequences within the same equivalence class, the usage of a rank-based distance over these representations helps to make the distance over sequences invariant.

Future work will aim at improving the proposed method by taking into account multiple hidden layers in the rank-based distance. On a more practical side, we will also aim at using this distance on music generation tasks in order to design algorithms capable of generating highly-structured melodies.

8

INTERACTIVE MUSIC GENERATION WITH UNARY CONSTRAINTS USING ANTICIPATION-RNNS

INTRODUCTION

We have seen in Ch. 4 that a number of powerful generative models on symbolic music have been proposed recently. If they now perform well on a variety of different musical datasets, from monophonic folk-music (Sect. 4.1.1.1) to polyphonic Bach chorales (Sect. 4.1.2.3), these models tend to face similar limitations: they do not provide musically-interesting ways for a user to interact with them. Most of the time, only an input seed can be specified in order to condition the model upon: once the generation is finished, the user can only accept the result or regenerate another musical content. As exposed in Sect. 3.4, we believe that this restriction hinders creativity since the user do not play an active part in the music creation process.

Generation in these generative models is often performed from left to right (Sect. 4.1.1); Recurrent Neural Networks (RNNs) [58] are generally used to estimate the probability of generating the next musical event, and generation is done by iteratively sampling one musical event after another. This left-to-right modeling seems natural since music unfolds through time and this holds both for monophonic [35, 145] and polyphonic [17, 93] music generation tasks. However, this does not match real compositional principles since composition is mostly done in an iterative and non-sequential way [11]. As a simple example, one may want to generate a melody that ends on a specific note, but generating such melodies while staying in the learned style (the melodies are sampled with the correct probabilities) is in general a non trivial problem when generation is performed from left to right. This problem has been solved when the generative model is a Markov model [119] but remains hard when considering arbitrary RNNs.

In order to solve issues raised by the left-to-right sampling scheme, approaches based on MCMC methods have been proposed, in the context of monophonic sequences with shallow models [136] or on polyphonic musical pieces using deeper models [62, 63]. If these MCMC methods allow to generate musically-convincing sequences while enforcing many user-defined constraints, the generation process is generally order of magnitudes longer than the simpler left-to-right generation scheme. This can prevent for instance using these models in real-time settings.

Another related approach is the one proposed in [88] where the authors address the problem of enforcing deterministic constraints

on the output sequences. Their approach relies on performing a gradient descent on a regularized objective that takes into account the amount of constraints that are violated in the output sequence. They start from a pre-trained unconstrained model and then "nudge" its weights until it produces a valid sequence. If their model is able to handle a wide range of constraints (such as requiring the output sequence to belong to a context-free language), it enforces these constraints using a costly procedure, namely Stochastic Gradient Descent (SGD). Sequences are generated using the deterministic argmax decoding procedure while our sampling scheme is non-deterministic, which we believe is an essential asset in the context of interactive music generation. The approach of [87] is similar to the latter approach in the sense that the authors enforce constraints via gradient descent. However, since they rely on convolutional restricted Boltzmann Machines, their sampling scheme is no longer deterministic. Their method is a way to sample polyphonic music having some imposed high-level structure (repetitions, patterns) which is imposed through the prescription of some predefined auto-correlation matrix.

The particularity of our approach is that it focuses on a smaller subset of constraints, namely unary constraints which allows our sampling scheme to be faster since the proposed model takes into account the set of constraints during the training phase instead of the generation phase.

Except from the approaches cited above, the problem of generating sequences while enforcing user-defined constraints is rarely considered in the general machine learning literature but it is of crucial importance when devising interactive generative models. In this article, we propose a neural network architecture called *Anticipation-RNN* which is capable of generating in the style learned from a database while enforcing user-defined unary constraints. Our model relies on two stacked RNNs, a *Constraint-RNN* going from right to left whose aim is to take into account future constraints and a *Token-RNN* going from left to right that generates the final output sequence. This architecture is very general and works with any RNN implementation. Furthermore, the generation process is fast as it only requires two neural network calls per musical event.

Even if the proposed architecture is composed of two RNNs going in opposite directions, it has not to be confused with the Bidirectional-RNNs (BRNNs) architectures [137] which are commonly used to either summarize an entire sequence as in [131] or in the context of supervised learning [60]. Even if there has been attempts to use BRNNs in an unsupervised setting [15], these methods are intrinsically-based on a MCMC sampling procedure which make them much slower than our proposed method. The idea of integrating future information to improve left to right generation using RNNs has been considered in the Variational Bi-LSTM architecture [139] or in the Twin Networks

architecture [138]. The aim of these architectures is to regularize the hidden states of the RNNs so that they better model the data distribution. If ideas could appear to be similar to the ones developed in this paper, these two approaches do not consider the generation of sequences under constraints but are a method to improve the existing RNNs architectures.

The plan for this article is the following: In Sect. 8.2, we precisely state the problem we consider and Sect. 8.3 describes the proposed architecture together with an adapted training procedure. Finally, we demonstrate experimentally the efficiency of our approach on the dataset of the *chorale melodies by J.S. Bach* in Sect. 8.4. In Sect. 8.5, we discuss about the generality of our approach and about future developments.

Code is available at https://github.com/Ghadjeres/Anticipation-RNN and the musical examples presented in this article can be listened to on the accompanying website: https://sites.google.com/view/anticipation-rnn-examples/accueil.

STATEMENT OF THE PROBLEM

We consider an i.i.d. dataset $\mathcal{D} := \{s = (s_1, \dots, s_N) \in \mathcal{A}^N\}$ of sequences of tokens $s_t \in \mathcal{A}$ of arbitrary length N over a vocabulary $\mathcal{A}$. We are interested in probabilistic models over sequences $p(s)$ such that

$$p(s) = \prod_t p(s_t|s_{<t}), \tag{43}$$

where

$$s_{<t} = \begin{cases} (s_1, \dots, s_{t-1}) & \text{for } t > 0 \\ \emptyset, & \text{if } t = 0. \end{cases} \tag{44}$$

This means that the generative model $p(s)$ over sequences is defined using the conditional probabilities $p(s_t|s_{<t})$ only. Generation with this generative model is performed iteratively by sampling s_t from $p(s_t|s_{<t})$ for $t = 1..N$ where N is arbitrary. Due to their simplicity and their efficiency, Recurrent Neural Networks (RNNs) are used to model the conditional probability distributions $p(s_t|s_{<t})$: they allow to reuse the same neural network over the different time steps by introducing a hidden state vector in order to summarize the previous observations we condition on. More precisely, by writing f the RNN, in_t its input, out_{t+1} its output and h_t its hidden state at time t, we have

$$out_{t+1}, h_{t+1} = f(in_t, h_t) \tag{45}$$

for all time indices t. When $in_t = s_t$, the vector out_{t+1} is used to define $p(s_{t+1}|s_{<t+1})$ for all time indices t without the need to take as an input the entire sequence history $s_{<t+1}$.

If this approach is successful on many applications, such a model can only be conditioned on the past which prevents some possible creative use for these models: we can easily fix the beginning $s_{<t}$ of a sequence and generate a continuation $s_{\geqslant t} = (s_t, \dots, s_N)$ but it becomes more intricate to fix the end $s_{\geqslant t}$ of a sequence and ask the model to generate a beginning sequence.

We now write $p_-(s)$ the probability of a sequence s when no constraint is set. For simplicity of notation, we will suppose that we only generate sequences of fixed length N and denote by $\mathcal{S} := \mathcal{A}^N$ the set of all sequences over $\mathcal{A}$. The aim of this article is to be able to enforce any set $\mathcal{C}$ of unary constraints given by:

$$\mathcal{C} = \{(i, c_i)\}_{i \in I}, \tag{46}$$

where I is the set of constrained time indexes and $c_i \in \mathcal{A}$ the value of the constrained note at time index i. Ideally, we want to sample constrained sequences

$$\mathcal{S}_+(\mathcal{C}) := \{s \in \mathcal{S}, \quad s_i = c_i \quad \forall (i, c_i) \in \mathcal{C}\} \tag{47}$$

with the "correct" probabilities. If we denote by $p_+(s|\mathcal{C})$ the probability of a sequence s in the constrained model conditioned on a set of constraints $\mathcal{C}$, this means that we want, for all set of constraints $\mathcal{C}$:

$$p_+(s|\mathcal{C}) = 0, \quad \forall s \notin \mathcal{S}_+(\mathcal{C}), \tag{48}$$

and

$$p_+(s|\mathcal{C}) = \frac{1}{\alpha} p_-(s), \quad \forall s \in \mathcal{S}, \tag{49}$$

where

$$\alpha := \sum_{s \in \mathcal{S}_+(\mathcal{C})} p_-(s).$$

To put it in words, each set of constraints $\mathcal{C}$ defines a subset $\mathcal{S}_+(\mathcal{C})$ of $\mathcal{S}$ from which we want to sample from using the probabilities (up to a normalization factor) given by p_-. However, sampling from $\mathcal{S}_+(\mathcal{C})$ using the acceptance-rejection sampling method is not efficient due to the arbitrary number of constraints. Exact sampling from $\mathcal{S}_+(\mathcal{C})$ is possible when the conditional probability distributions are modeled using models such as Markov models but is intractable in general. This problem in the case of Markov models can in fact be exactly solved when considering more complex constraints on the space of sequences such as imposing the equality or the difference between two sequences symbols s_i and s_j. Generalizations of this problem to other types of constraints are discussed in Sect. 8.5.

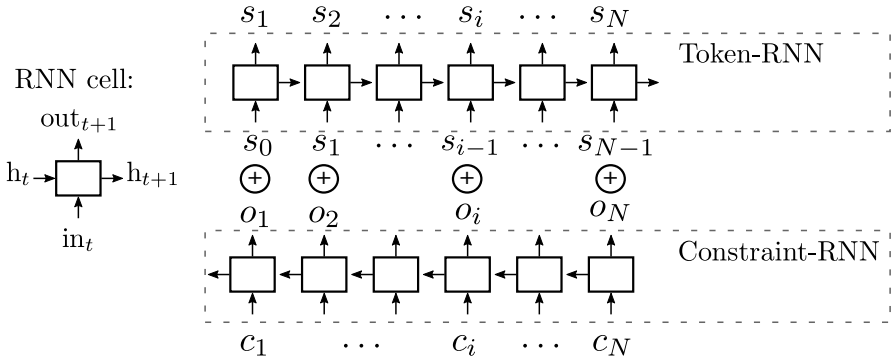


Figure 38: Anticipation-RNN architecture. The aim is to predict $(s_1, \dots, s_N)$ given $(c_1, \dots, c_N)$ and $(s_0, \dots, s_{N-1})$.

THE MODEL

The problem when trying to enforce a constraint $c := (i, c_i)$ is that imposing such a constraint on time index i "twists" the conditional probability distributions $p_-(s_t|s_{<t})$ for $t < i$. However, the direct computation of $p_-(s_t|s_{<t}, s_i = c_i)$ (using Bayes rule when only $p_-(s_t|s_{<t})$ is known) is computationally expensive.

The idea to overcome this issue is to introduce a neural network in order to summarize the set of constraints $\mathcal{C}$. To this end, we introduce an additional token `NC` (No Constraint) to $\mathcal{A}$ indicating that no unary constraint is set at a given position. By doing this, we can rewrite the set $\mathcal{C}$ as a sequence $c = (c_1, \dots, c_N)$ where $c_i \in \mathcal{A} \cup \{\texttt{NC}\}$. We then introduce a RNN called *Constraint-RNN* in order to summarize the sequence of all constraints. This RNN goes backward (from c_N to c_1) and *all* its outputs are used to condition a second RNN called *Token-RNN*.

This architecture, called *Anticipation-RNN* since the Token-RNN is conditioned on what may come next, is depicted in Fig. 38. We notated by $(o_1, \dots, o_N)$ the output sequence of the Constraint-RNN (for notational simplicity, we reversed the sequence numbering: the first output of the Constraint-RNN is o_N in our notation). The aim of the output vector o_t is to summarize all information about constraints from time t up to the end of the sequence. This vector is then concatenated to the input s_{t-1} of the Token-RNN at time index t whose aim is to predict s_t.

Basically, this amounts to modeling the conditional probability distribution $p_+(s|c)$ using the following factorization:

$$p_+(s|c) = \prod_t p_+(s_t|s_{<t}, c_{\geqslant t}) = \prod_t p_+(s_t|s_{<t}, o_t), \tag{50}$$

where $c_{\geqslant t}$ is defined similarly as in (44).

Our approach differs from the approaches using Markov models in the sense that we directly train the conditional probability distribution (50) rather than trying to sample sequences in $\mathcal{S}_+(\mathcal{C})$ using p_-:

we want our probabilistic model to be able to directly enforce hard constraints.

The Anticipation-RNN thus takes as an input both a sequence of tokens $(s_0, \dots, s_{N-1})$ and a sequence of constraints $(c_1, \dots, c_N)$ and has to predict the shifted sequence $(s_1, \dots, s_N)$. The only requirement here is that the constraints have to be coherent with the sequence: $c_i = s_i$ if $c_i \neq \texttt{NC}$. Since we want our model to be able to deal with any unary constraints, we consider the dataset of couples of token-sequences and constraint-sequences $\mathcal{D}_+$ such that

$$\mathcal{D}_+ := \left\{(s, m(s)), \quad \forall s \in \mathcal{D}, \forall m \in \{0,1\}^N\right\}, \tag{51}$$

where $\{0,1\}^N$ is the set of all binary masks: the sequence of constraints $m(s)$ is then defined as the sequence $(c_1, \dots, c_N)$ where $c_i = s_i$ if $m_i = 1$ and $c_i = \texttt{NC}$ otherwise.

It is important to note that this model is able to handle not only unary constraints, but can also include additional metadata information about the sequence of tokens whose changes we have to anticipate. Indeed, by including such temporal information in the c variables, this model can then learn to anticipate how to generate the tokens that will lead to a sequence complying with the provided metadata in a smooth way. These metadata can be musically-relevant features such as the current key or mode, or the position of the cadences as it is done in [62].

This sampling procedure is fast since it only needs two RNN passes on the sequence. This modeling is thus particularly well-suited for the real-time interactive generation of music. Furthermore, once the output of the Constraint-RNN o is computed, sampling techniques usually applied in sequence generation tasks such as beam search [17, 156] can be used without additional computing costs.

EXPERIMENTAL RESULTS

Dataset preprocessing

We evaluated our architecture on the dataset of the melodies from the four-part chorale harmonizations by J.S. Bach. This dataset is available in the music21 Python package [38] and we extracted the soprano parts from all 402 chorales that are in 4/4. In order to encode these monophonic sequences, we used the melodico-rhythmic encoding described in [62]. In this encoding, time is quantized using a sixteenth note as the smallest subdivision (each beat is divided into four equal parts). On each of these subdivisions, the real name of the note is used as a token if it is the subdivision on which the note is played, otherwise, an additional token denoted as "__" is used in order to indicate that the current note is held. A "rest" token is also used in

```
D4 __ E4 __ A4 __ __ __ G4 __ F#4 __ E4 __ __ __
```

Figure 39: Melodico-rhythmic encoding of the first bar of the melody of Fig. 45a. Each note name such as D4 or F#4 is considered as a single token.

order to handle rests. An example of an encoded melody using this encoding is displayed in Fig. 39.

The advantage of using such an encoding is that it allows to encode a monophonic musical sequence using only one sequence of tokens. Furthermore, it does not rely on the traditional MIDI pitch encoding but on the real note names: among other benefits, this allows to generate music sheets which are immediately readable and understandable by a musician and with no spelling mistakes. From a machine learning perspective, this has the effect of implicitly taking into account the current key and not throwing away this important piece of information. The model is thus more capable of generating coherent musical phrases. A simple example for this is that this encoding helps to distinguish between a E# and a F by considering them as two different notes. Indeed, these two notes would appear in contexts that are in different keys (in C# major or F# minor in the first case, in C major or F major in the second case for instance).

We also perform data augmentation by transposing all sequences in all possible keys as long as the transposed sequence lies within the original voice range. We end up with an alphabet of tokens $\mathcal{A}$ of size 125.

Implementation details

We used a 2-layer stacked LSTM [69] for both the Constraint-RNN and the Token-RNN using the PyTorch [125] deep learning framework. Both LSTM networks have 256 units and the constraints tokens c_i and the input tokens s_i are embedded using the same embedding of size 20. Sequences are padded with `START` and `END` symbols so that the model can learn when to start and when to finish. We add dropout on the input and between the LSTM layers and discuss the effect of the choice of these hyperparameters in Sec. 8.4.3. We found that adding input on the input is crucial and set this value to 20%..

We fixed the length of the training sub-sequences to be 20-beat long which means that, using the encoding described in Sect. 8.4.1 that we consider sequences of tokens of size 80. The network is trained to minimize the categorical cross-entropy between the true token at position 40 and its prediction. For each training sequence, we sample the binary masks $m(s)$ of (51) by uniformly sampling a masking ratio $p \in [0, 1]$ and then setting each unary constraint with probability p.

We perform stochastic gradient descent using the Adam algorithm [79] using the default settings provided by PyTorch for 10 epochs

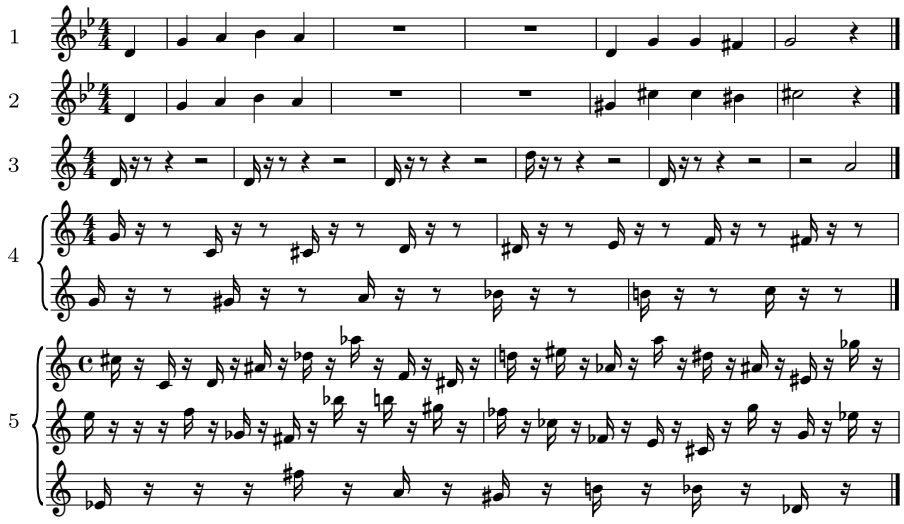


Figure 40: Sets of constraints $\mathcal{C}^i$ described in Sec. 8.4.3, for i ranging from 1 to 5. In this particular figure, rests denote the absence of constraints.

with a batch size of 256. In this setting, our best model achieves a validation accuracy of 92.9% with a validation loss of 0.22. These figures are of course highly-dependent on our modeling choices such as the number of subdivision per beat, the preprocessing of our corpus as well as the way we sampled the binary masks.

The sampling procedure is then done iteratively from left to right by sampling the token at time t according to the probabilities given by $p_+(s_t|s_{<t}, o_t)$, where $s_{<t}$ is the sequence of previously generated tokens and o_t the output of the Constraint-RNN at position t.

Enforcing the constraints

We first check that the proposed architecture is able to enforce unary constraints, namely, that it fulfills the requirement (48).

In order to evaluate this property, we compute the amount of constraints that are enforced for various set of constraints $\mathcal{C}$. We chose for these sets of constraints different "kinds" of constraints, from constraints that are in the "style of the corpus" to constraints that are totally "out-of-style". More precisely, we considered:

- $\mathcal{C}^1$: the beginning and the ending of an existing chorale melody (first five bars of the chorale melody "Wer nur den lieben Gott läßt walten" with two ablated bars),
- $\mathcal{C}^2$: the beginning and the ending of the same chorale melody, but where the ending has been transposed to a distant key (from G minor to C# minor),
- $\mathcal{C}^3$: constraints forcing the model to make "big" leaps (chorale melodies tend to be composed of conjunct melodic motions),

- $\mathcal{C}^4$: a chromatic ascending scale,
- $\mathcal{C}^5$: random notes every eighth note,
- $\mathcal{C}^6$: the same random notes as above, but every quarter note.

These set of unary constraints are displayed in Fig. 40.

We measure the influence of the amount of the dropout that we use in our models (dropout between the LSTM layers) on the following task: for each set of constraints $\mathcal{C}^i$ and for each model, we generate 1000 sequences using $p_+(.|\mathcal{C}^i)$ and compute the percentage of constrained notes that are sampled correctly. We report the results in Tab. 2.

	$\mathcal{C}^1$	$\mathcal{C}^2$	$\mathcal{C}^3$	$\mathcal{C}^4$	$\mathcal{C}^5$	$\mathcal{C}^6$
LSTM dropout=0.2	99.78	99.73	98.78	99.77	41.28	57.06
LSTM dropout=0.5	99.90	99.01	99.32	98.68	43.83	62.08

Table 2: Percentage of correctly-sampled constrained notes for different models p_+ differing only by the amount of dropout they use and constrained on different sets of constraints.

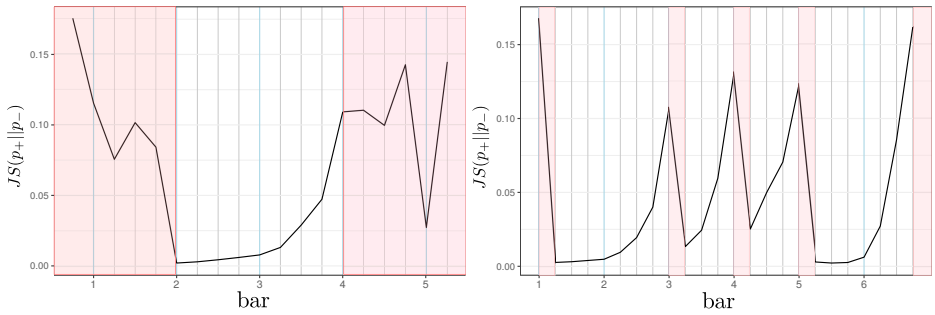
These results show that for all sets of constraints that define a "possibly-in-style" musical constraint (constraint sets $\mathcal{C}^1$ to $\mathcal{C}^4$), the model manages to enforce the constraints efficiently: even if such constraints could not be encountered in the original dataset (constraint sets $\mathcal{C}^2$ and $\mathcal{C}^4$) . On the contrary, for truly out-of-style constraints (constraint sets $\mathcal{C}^5$ and $\mathcal{C}^6$), the model performs poorly on the task of enforcing these constraints. We do not think that it is a drawback of the model since its aim is to generate melodies in the style of the corpus which is made impossible when constrained with these incoherent constraints.

Table. 2 also reveals the non-trivial effects of the choice of the amount of dropout of the models upon their performance on this task.

Anticipation capabilities

If the preceding section demonstrated that the Anticipation-RNN architecture is able to enforce a wide variety of sets of unary constraints, we will explore in this section the role of the Constraint-RNN and in particular how it is able to learn how to "propagate" the constraints backwards, making the Token-RNN able to anticipate what will come next.

For this, we will evaluate how the constrained model deviates from the unconstrained model. We compare the constrained model p_+ on the same set of constraints $\mathcal{C}$ with its unconstrained counterpart p_-.



(a) Set of constraints $\mathcal{C}^1$ (b) Set of constraints $\mathcal{C}^3$

Figure 41: Plot of the evolution of the Jensen-Shannon divergence $JS(p_+(s_t|s_{i<t}, \mathcal{C}^i)||p_-(s_t|s_{i<t}))$ between the constrained model p_+ and the unconstrained model p_- during generation for two sets of constraints $\mathcal{C}^1$ and $\mathcal{C}^3$. Each point represents the average value of the divergence over one beat. The location of the constraints has been highlighted in red.

The latter is obtained by conditioning the model of Fig. 38 on a sequence of constraints in the special case where no constraint is set: the sequence of constraints is (`NC`, ..., `NC`).

More precisely, for a set of constraints $\mathcal{C}$, we quantify how the probability distributions $p_+(.|s_{<t}, \mathcal{C})$ differ from the probability distributions $p_-(.|s_{<t})$ by computing how dissimilar they are. We chose as a measure of dissimilarity [12, 43] the Jensen-Shannon divergence [6, 112] which is defined by:

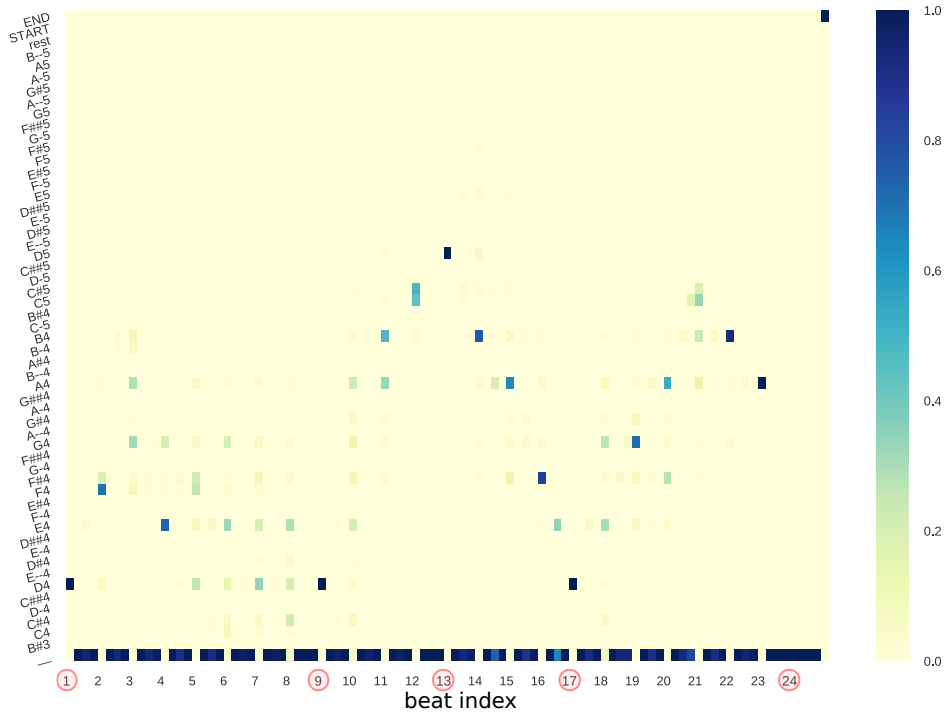
$$JS(p||q) := \frac{1}{2}KL(p||m) + \frac{1}{2}KL(q||m), \tag{52}$$

where $m = \frac{p+q}{2}$, with KL denoting the Kullback-Leibler: divergence

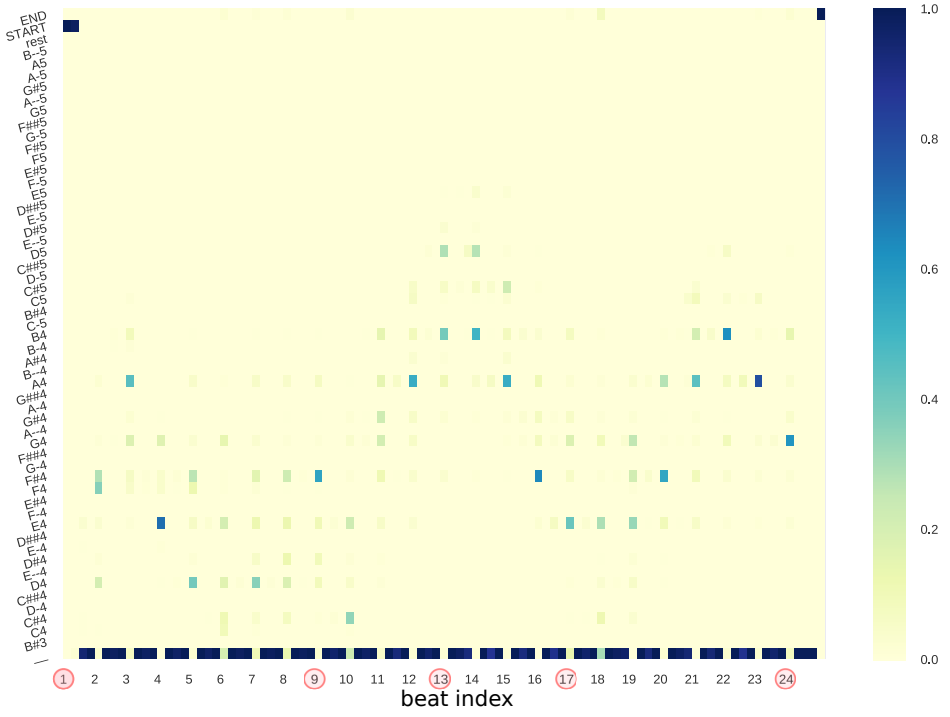
$$KL(p||q) := \sum_i p_i \log\left(\frac{p_i}{q_i}\right). \tag{53}$$

The Jensen-Shannon divergence has the property of being symmetric, bounded (and thus always definite) and its square root satisfies the triangle inequality [115] which is an important feature compared to other divergences.

In Fig. 41 we plot the evolution of the between the two distributions $p_+(.|s_{<t}, \mathcal{C})$ and $p_-(.|s_{<t})$ during generation for different set of constraints $\mathcal{C}$. We generated 1000 sequences using the constrained model and computed the average Jensen-Shannon divergence between the two models for each time step. We then averaged the values over each beat in order not to take into account the intra-beat variations. Indeed, due to encoding we chose as well as to the singularity of the musical data we considered, patterns of oscillations appear. Indeed, both models agree in putting much of their probability mass on the hold symbol "__" on the second sixteenth note of each beat since the



(a) Constrained case: $p = p_+(.|\mathcal{C}^3)$



(b) Unconstrained case: $p = p_-$

Figure 42: Plot of $p(s_t|s_{<t})$ as a function of t during the generation of the melody displayed in Fig. 45a in the constrained and unconstrained cases. Beats on which a constraint is set are circled.

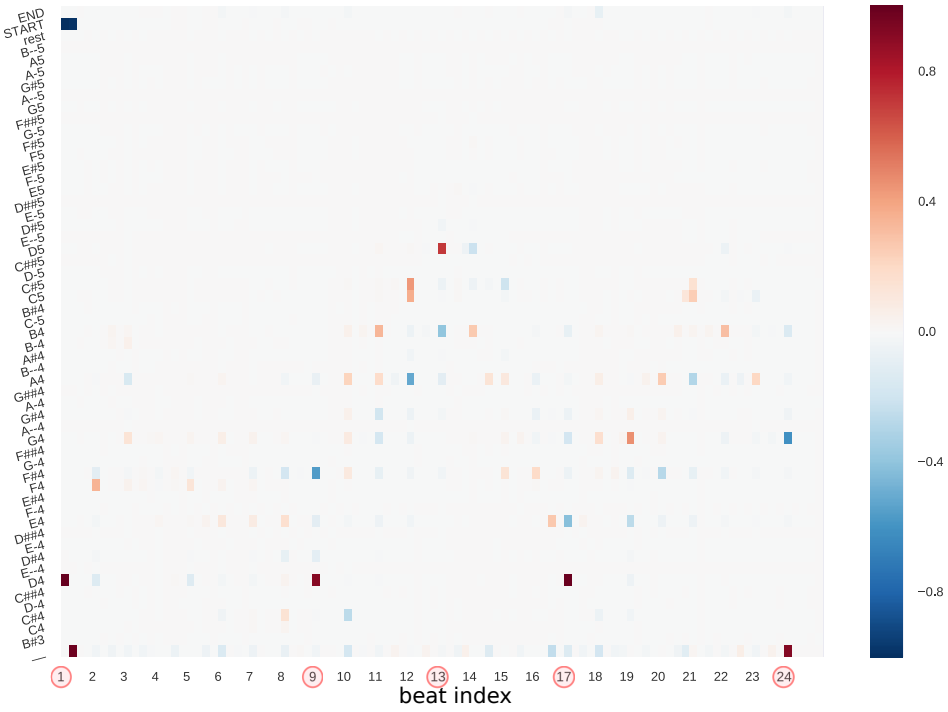


Figure 43: Difference between $p_+(s_t|s_{<t})$ and $p_-(s_t|s_{<t})$ as a function of t during the generation of the melody displayed in Fig. 45a. Beats on which a constraint is set are circled. We see that between beats 9 to 13, the probability mass of the constrained model p_+ is shifted upwards (compared to the probability distribution given by the unconstrained model p_-) in order to enforce the unary constraint D5 set at beat 13 . From beat 13 to 17, the situation is reversed: the probability mass of the constrained model p_+ is shifted downwards in order to enforce the unary constraint D4 on beat 17.

soprano parts in Bach chorales are mostly composed of half notes, quarter notes and eighth notes. This is independent of the presence or absence of constraints so the constrained and unconstrained models make similar predictions on these time steps resulting in a low divergence.

This plot confirms that the constraints are propagated backwards in time and the Token-RNN is not only able to enforce constraints but also to anticipate how to do so.

We now illustrates this feature on a specific example. Figure 42 shows the evolution of $p_+(s_t|s_{<t}, \mathcal{C}^3)$ and $p_-(s_t|s_{<t})$ during generation. It is interesting to note that the conditional probability distributions returned by $p_+(s_t|s_{<t}, \mathcal{C}^3)$ are more concentrated on specific values than the ones returned by $p_-(s_t|s_{<t})$. The concentration of the all probability mass of $p_+(s_t|s_{<t}, \mathcal{C}^3)$ on constrained notes confirms, on this specific example, that the proposed architecture has learned to enforce hard unary constraints.

In order to understand the effect of the constraints, we plot the difference between the two distributions of Fig. 42 for each time step in Fig. 43. This highlights the fact that the probability mass distribution of p_+ is "shifted upwards" few beats in advance when the next unary constraint is higher than the current note, and "downwards" in the opposite case.

Sampling with the correct probabilities

We now evaluate that the sampling using p_+ fulfills the requirement (49). This means that for any set of constraints $\mathcal{C}$, the ratio between the probabilities of two sequences in $\mathcal{S}_+(\mathcal{C})$ is identical if probabilities are computed using the unconstrained model $p_-(.)$ or if they are computed using the constrained model $p_+(.|\mathcal{C})$. We introduce the set of constraints $\mathcal{C}^0$ consisting of a single constrained note.

For a given set of constraints $\mathcal{C}$, we generated 500 sequences and verified that the requirement (48) is fulfilled for all of these sequences (i.e. all constraints are enforced). In order to check the fulfillment of the requirement (49), we plot for each sequence s its probability in the constrained model $p_+(s)$ (defined as in Eq. 43) as a function of $p_-(s)$ in logarithmic space. We compute these probabilities using (50), but only kept the time steps on which notes are not constrained. The resulting plots are shown in Fig. 44. Table. 3 quantifies to which amount the two distributions are proportional on the subsets $\mathcal{S}_+(\mathcal{C}^i)$ for different set of constraints $\mathcal{C}^i$ and for different models.

The translation in logarithmic space indicates the proportionality between the two distributions as desired.

The conclusion is that our model is able to correctly enforce all constraints for sets of constraints that are plausible with respect to the training dataset (Sect. 8.4.3), and that on these specific sets of

	$\mathcal{C}^0$	$\mathcal{C}^1$	$\mathcal{C}^2$	$\mathcal{C}^3$
LSTM dropout=0.2, dropout on input=0.2	0.99	0.99	1.05	0.96
LSTM dropout=0.5, dropout on input=0.2	0.99	0.93	1.00	0.92

Table 3: Slopes of the linear interpolations displayed in Fig. 44 for different models and different set of constraints $\mathcal{C}^i$. The closer the values are to one, the better the requirement (49) is achieved.

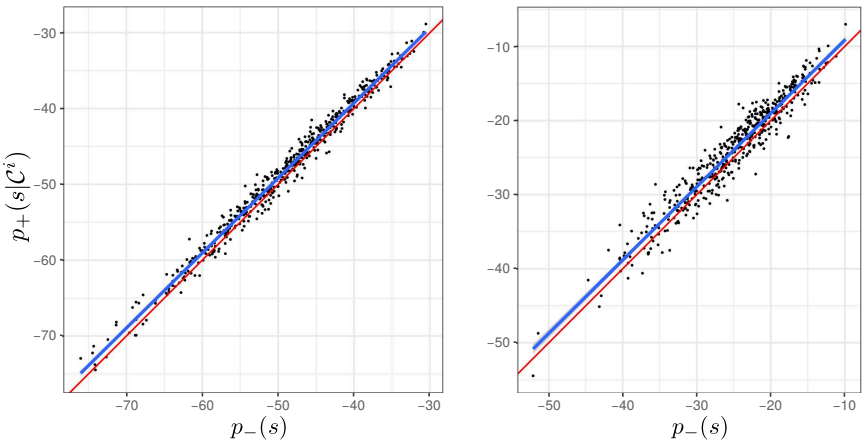


Figure 44: Point plots in logarithmic scale of $\ln p_+(s|\mathcal{C}^i)$ (y-axis) versus $p_-(s)$ (x-axis) on a set of 500 sequences generated using $p_+(s|\mathcal{C}^i)$, for $\mathcal{C}^0$ and $\mathcal{C}^3$. The identity map is displayed in red and the linear regression of the data points in blue. The lines are closed to being parallel indicating the proportionality between the two distributions.

constraints, our sampling procedure respects the relative probabilities between the sequences. In other words, the Anticipation-RNN is able to sample with the correct probabilities a subset of sequences defined by a set of unary constraints.

Musical examples

We end this section with the discussion over some generated constrained sequences. Figure 45 shows examples of the enforcement and the propagation of the constraints for the set of constraints $\mathcal{C}^3$: even if generation is done from left to right, the model is able to generate compelling musical phrases while enforcing the constraints. In particular, we see that the model is able to "anticipate" the moment when it has to "go" from a low-pitched note to a high-pitched one and vice versa. The use of the melodico-rhythmic encoding allows to only impose that a note should be played at a given time, without specifying its rhythm. It is interesting to note that such a wide melodic contour (going from a D4 to a D5 and then going back to

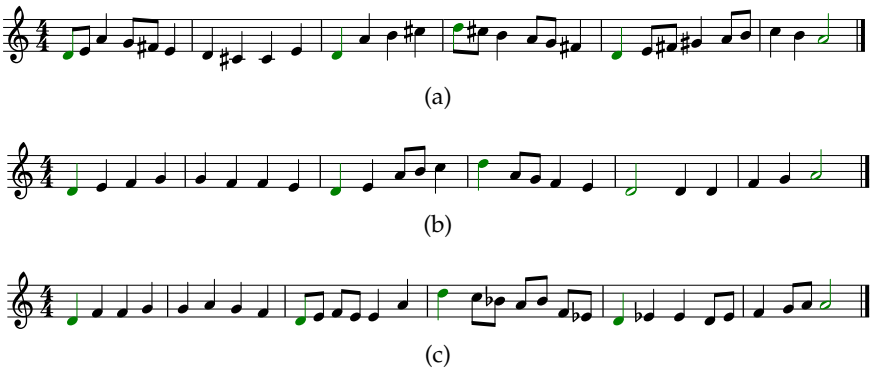


Figure 45: Examples of generated sequences in the style of the soprano parts of the J.S. Bach chorales. All examples are subject to the same set of unary constraints $\mathcal{C}^3$ which is indicated using green notes.

a D4 in only two bars) is unusual for a chorale melody. Nonetheless, the proposed model is able to generate a convincing Bach-like chorale melody. The three displayed examples show that there is a great variability in the generated solutions: even when constrained on the same set of constraints, the generated melodies have distinct characteristics such as, for example, the key they are in or where cadences could be.

Similarly to [62], we provide a plugin for the MuseScore music score editor which allows to call the Anticipation-RNN in an intuitive way.

CONCLUSION

We presented the Anticipation-RNN, a simple but efficient way to generate sequences in a learned style while enforcing unary constraints. This method is general and can be used to improve many existing RNN-based generative models. Contrary to other approaches, we teach the model to learn to enforce hard constraints at training time. We believe that this approach is a first step towards the generation of musical sequences subjected to more complex constraints.

The constrained generation procedure is fast since it requires only 2N RNN calls, where N is the length of the generated sequence; as it does not require extensive computational resources and provides an interesting user-machine interaction, we think that this architecture paves the way to the development of creative real-time composition software. We also think that this fast sampling could be used jointly with MCMC methods in order to provide fast initializations.

Our approach can be seen as a general way to condition RNN models on time-dependent metadata. Indeed, the variable c in (50) is not only restricted to the value of the unary constraints, but can contain more information such as the location of the cadences or the current key. We successfully applied the Anticipation-RNN in this setting and report that it manages to enforce these interesting and natural musical

constraints in a smooth way while staying in the style of the training corpus.

Future work will aim at handling other types of constraints (imposing the rhythm of the sequences, enforcing the equality between two notes or introducing soft constraints) and developing responsive user interfaces so that all the possibilities offered by this architecture can be used by a wide audience.

9

GLSR-VAE: GEODESIC LATENT SPACE REGULARIZATION FOR VARIATIONAL AUTOENCODER ARCHITECTURES

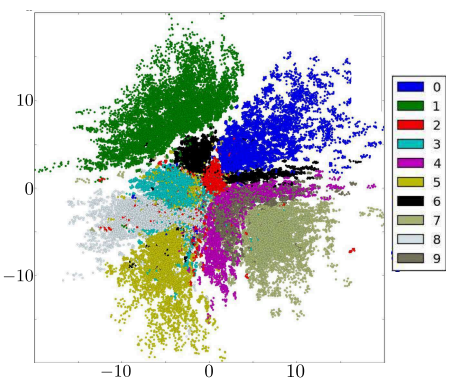
INTRODUCTION

Autoencoders [14] are useful for learning to encode observable data into a latent space of smaller dimensionality. They perform dimensionality reduction (manifold learning) and can be seen as a non-linear generalization of the Principal Component Analysis [39]. However, the latent variable space often lacks of structure [153] and it is impossible, by construction, to sample from the data distribution. The Variational Autoencoder (VAE) framework [78] addresses these two issues by introducing a regularization on the latent space together with an adapted training procedure. This allows to train complex generative models with latent variables while providing a way to sample from the learned data distribution, which makes it useful for unsupervised density modeling.

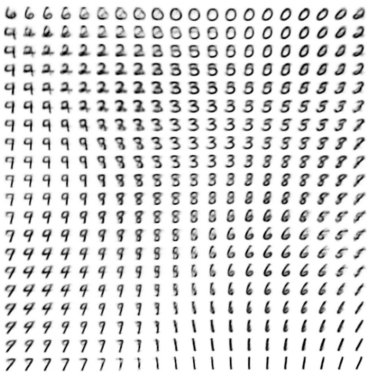
Once trained, a VAE provides a decoding function, i.e. a mapping from a low-dimensional latent space to the observation space which defines what is usually called a *data manifold* (see Fig. 46b). It is interesting to see that, even if the observation space is discrete, the latent variable space is continuous which allows one to define *continuous paths* in the observation space i.e. images of continuous paths in the latent variable space. This interpolation scheme has been successfully applied to image generation [61, 98] or text generation [18]. However, any continuous path in the latent space can produce an interpolation in the observation space and there is no way to prefer one over another *a priori*; thus the straight line between two points in latent space will not necessary produce the "best" interpolation in data space.

When dealing with data that contains more information, such as labels or interpretive quantities, it is interesting to see if and how this information has been encoded into the latent space (see Fig. 46a). Understanding the latent space structure can be of great use for the generation of new content as it can provide a way to manipulate high-level concepts in a creative way.

One way to control the generating process on annotated data is by conditioning the VAE model, resulting in the Conditional Variational AutoEncoder (CVAE) architectures [141, 162]. These models can be used to generate images with specific attributes but also allow to generate interpolation between images by changing only a given attribute. In these approaches, the CVAE latent spaces do not con-



(a) VAE encoding of the MNIST dataset in a latent variable space $\mathcal{Z} := \mathbf{R}^2$. Each point corresponds to a dataset point and colors. Reproduced from [98].



(b) Visualization of the learned data manifold. Reproduced from [78].

Figure 46: Two visualizations of the latent space of a VAE trained on the MNIST dataset.

tain the high-level information but the randomness of the produced images for fixed attributes.

Another approach, where there is no decoupling between the attributes and the latent space representation, consists in finding *attribute vectors* which are vectors in the latent space that could encode a given high-level feature or concept. Translating an encoded input vector by an attribute vector before decoding it should ideally add an additional attribute to the data. This has been successfully used for image generation with VAEs (coupled with an adversarially learned similarity metric) in [86] and even directly on the high-level feature representation spaces of a classifier [152]. However, these approaches rely on finding *a posteriori* interpretations of the learned latent space and there is no theoretical reason that simple vector arithmetic has a particular significance in this setting (even if it has proved successful in real applications like word2vec [104]). Indeed, in the original VAE article [78] the MNIST manifold (reproduced in Fig. 46b) has been obtained by transforming a linearly spaced coordinate grid on the unit square through the inverse CDF of the normal distribution in order to obtain "equally-probable" spaces between each decoded image. This advocates for the fact that the latent space coordinates of the data manifold need not be *perceived* as geodesic normal coordinates. That is, decoding a straight line drawn in the latent space does not give rise to elements whose attributes vary uniformly.

In this chapter, we focus on fixing *a priori* the geometry of the latent space when the dataset elements possess continuous (or discrete and *ordered*) attributes. By introducing a *Geodesic Latent Space Regularization* (GLSR), we show that it is possible to relate variations in the latent space to variations of the attributes of the decoded elements.

Augmenting the VAE training procedure with a regularizing term has been recently explored in [84] in the context of image generation where the introduction of a *discriminative regularization* is aimed at improving the visual quality of the samples using a pre-trained classifier. Our approach differs from the one above in the fact that the GLSR favors latent space representations with fixed attribute *directions* and focuses more on the latent space structure.

We show that adding this regularization grants the latent space a meaningful interpretation while retaining the possibility to sample from the learned data distribution. We demonstrate our claim with experiments on musical data. Our experiments suggest that this regularization also helps to learn latent variable spaces with little correlation between regularized and non regularized dimensions. Adding the possibility to *gradually* alter a generated sample according to some user-defined criteria can be of great use in many generative tasks; it introduces a new human-machine interaction that is accordance with the objectives described in Sect. 3.4. Since decoding is fast, we believe that this technique can be used in real time for interactive and creative purposes in many interesting and novel ways.

REGULARIZED VARIATIONAL AUTOENCODERS

Background on Variational Autoencoders

We define a *Variational AutoEncoder* (VAE) as a deep generative model (like Generative Adversarial Networks (GANs) [70]) for observations $x \in \mathcal{X}$ that depends on latent variables $z \in \mathcal{Z}$ by writing the joint distribution $p_\theta(x, z)$ as

$$p_\theta(x, z) = p(z)p_\theta(x|z),$$

where $p(z)$ is a *prior* distribution over z and $p_\theta(x|z)$ a *conditional distribution* parametrized by a neural network $\mathrm{NN}(\theta)$. Given a i.i.d. dataset $X = \{x^1, \ldots, x^N\}$ of elements in $\mathcal{X}$, we seek the parameter θ maximizing the dataset likelihood

$$\log p_\theta(X) = \sum_{i=1}^{N} \log p_\theta(x^i). \tag{54}$$

However, the marginal probability

$$p_\theta(x) = \int p_\theta(x, z)dz$$

and the posterior probability

$$p_\theta(z|x) = \frac{p_\theta(x, z)}{p_\theta(x)} = \frac{p_\theta(x, z)}{\int p_\theta(x, z)dz}$$

are generally both computationally intractable which makes maximum likelihood estimation unfeasible. The solution proposed in [78] consists in preforming Variational Inference (VI) by introducing a parametric variational distribution $q_\phi(z|x)$ to approximate the model's posterior distribution and lower-bound the marginal log-likelihood of an observation x; this results in:

$$\begin{aligned}\log p_\theta(x) \geqslant & \\ & \mathbf{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - D_{KL}(q_\phi(z|x)||p(z)) \\ & := \mathcal{L}(x;\theta,\phi),\end{aligned} \tag{55}$$

where D_{KL} denotes the Kullback-Leibler divergence [37].

Training is performed by maximizing the *Evidence Lower BOund* (ELBO) of the dataset

$$\mathcal{L}(\theta,\phi) := \sum_{i=1}^{N} \mathcal{L}(x^i;\theta,\phi) \tag{56}$$

by jointly optimizing over the parameters θ and ϕ. Depending on the choice of the prior $p(z)$ and of the variational approximation $q_\phi(z|x)$, the Kullback-Leibler divergence $D_{KL}(q(z|x)||p(z))$ can either be computed analytically or approximated with Monte Carlo integration.

Equation 55 can be understood as an autoencoder with stochastic units (first term of $\mathcal{L}(x;\theta,\phi)$) together with a regularization term given by the Kullback-Leibler divergence between the approximation of the posterior and the prior. In this analogy, the distribution $q_\phi(z|x)$ plays the role of the *encoder* network while $p_\theta(x|z)$ stands for the *decoder* network.

Geodesic Latent Space Regularization (GLSR)

We now suppose that we have access to additional information about the observation space $\mathcal{X}$, namely that it possesses ordered quantities of interest that we want to take into account in our modeling process. These quantities of interest are given as K independent differentiable real *attribute functions* $\{g_k\}$ on $\mathcal{X}$, with K less than the dimension of the latent space.

In order to better understand and visualize what a VAE has learned, it can be interesting to see how the expectations of the attribute functions

$$G_k : z \mapsto \mathbf{E}_{p_\theta(x|z)}[g_k(x)] \tag{57}$$

behave as functions from $\mathcal{Z}$ to $\mathbf{R}$. In the Information Geometry (IG) literature [5, 6], the G_k functions are called the *moment parameters* of the statistics g_k.

Contrary to other approaches which try to find *attribute vectors* or *attribute directions a posteriori*, we propose to impose the directions of

interest in the latent space by linking changes in the latent space $\mathcal{Z}$ to changes of the G_k functions at training time. Indeed, linking changes of G_k (that have meanings in applications) to changes of the latent variable z is a key point for steering (interactively) generation.

This can be enforced by adding a regularization term over $z = (z_1, \ldots, z_{\dim \mathcal{Z}})$ to the ELBO $\mathcal{L}(\theta, \phi)$ of Eq. 56. We define the *Geodesic Latent Space Regularization for the Variational Auto-Encoder* (GLSR-VAE) by

$$\mathcal{R}_{\text{geo}}(z; \{g_k\}, \theta) := \sum_{k=1}^{K} \mathcal{R}_k(z; \theta) \tag{58}$$

where

$$\mathcal{R}_k(z; \theta) = \log r_k \left(\frac{\partial G_k}{\partial z_k}(z) \right). \tag{59}$$

The distributions r_k over the values of the partial derivatives of G_k are chosen so that $\mathbf{E}_u[r_k(u)] > 0$, and preferably peaked around its mean value (small variance). Their choice is discussed in Sect. 9.5.

Ideally (in the case where the distributions r_k are given by Dirac delta functions with strictly positive means), this regularization forces infinitesimal changes dz_k of the variable z to be proportional (with a positive factor) to infinitesimal changes of the functions G_k (Eq. 72). In this case, for $z_{K+1}, \ldots, z_{\dim \mathcal{Z}} \in \mathcal{Z}$ fixed, the mapping

$$(z_1, \ldots, z_K) \mapsto (G_1(z), \ldots, G_K(z)) \in \mathbf{R}^K, \tag{60}$$

where $z = (z_1, \ldots, z_K, z_{K+1}, \ldots, z_{\dim \mathcal{Z}})$ defines a Euclidean manifold in which geodesics are given by all straight lines.

To summarize, we are maximizing the following regularized ELBO:

$$\mathcal{L}_{\text{reg}}(x; \theta, \phi) := \mathbf{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z) + \mathcal{R}_{\text{geo}}(z; \{g_k\}, \theta) \right] - D_{KL}(q_\phi(z|x) \| p(z)). \tag{61}$$

Note that we are taking the expectation of $\mathcal{R}_{\text{geo}}$ with respect to the variational distribution $q_\phi(z|x)$.

Equation (61) can be maximized with stochastic gradient ascent using Monte-Carlo estimates of the intractable estimations. We also use the reparametrization trick [78] on stochastic variables to obtain low-variance gradients.

EXPERIMENTS

In this section, we report experiments on training a VAE on the task of modeling the distribution of *chorale melodies in the style of J.S. Bach* (see Sect. 2.3.4) with a geodesic latent space regularization. Learning

good latent representations for discrete sequences is known to be a challenging problem with specific issues (compared to the continuous case) as pinpointed in [75]. Sect. 9.3.1 describes how we used the VAE framework in the context of sequence generation, Sect. 9.3.2 exposes the dataset we considered and Sect. 9.3.3 presents experimental results on the influence of the geodesic latent space regularization tailored for a musical application. A more detailed account on our implementation is deferred to Sect. 9.4.

VAEs for Sequence Generation

We focus in this paper on the generation of discrete sequences of a given length using VAEs. Contrary to recent approaches [31, 47, 49], we do not use recurrent latent variable models but encode each entire sequence in a single latent variable.

In this specific case, each sequence $x = (x_1, \dots, x_T) \in \mathcal{X}$ is composed of T time steps and has its elements in $[A]$, where A is the number of possible tokens while the variable z is a vector in $\mathcal{Z}$.

We choose the prior $p(z)$ to be a standard Gaussian distribution with zero mean and unit variance.

The approximated posterior or encoder $q_\phi(z|x)$ is modeled using a normal distribution $\mathcal{N}(\mu(x), \sigma^2(x))$ where the functions μ and σ^2 are implemented by Recurrent Neural Networks (RNNs) [58].

When modeling the conditional distribution $p_\theta(x|z)$ on sequences from $\mathcal{X}$, we suppose that all variables x_i are independent, which means that we have the following factorization:

$$p_\theta(x|z) := \prod_{i=1}^{T} p_\theta^i(x_i|z). \tag{62}$$

In order to take into account the sequential aspect of the data and to make our model size independent of the sequence length T, we implement $p_\theta(x|z)$ using a RNN. The particularity of our implementation is that the latent variable z is only passed as an input of the RNN decoder on the first time step. To enforce this, we introduce a binary mask $m \in \{0, 1\}^T$ such that $m_1 = 1$ and $m_i = 0$ for $i > 1$ and finally write

$$p_\theta(x|z) := \prod_{i=1}^{T} p_\theta^i(x_i|m_i * z, m_{<i}), \tag{63}$$

where the multiplication is a scalar multiplication and where $m_{<i} := \{m_1, \dots, m_{i-1}\}$ for $i > 1$ and is $\emptyset$ for $i = 1$. In practice, this is implemented using one RNN cell which takes as input $m_i * z, m_i$ and the previous hidden state h_{i-1}. The RNN takes also the binary mask itself as an input so that our model differentiates the case $z = 0$ from the case where no latent variable is given.

The decoder $p_\theta(x|z)$ returns in fact probabilities over $\mathcal{X}$. In order to obtain a sequence in $\mathcal{X}$ we have typically two strategies which are: taking the maximum a posteriori (MAP) sequence

$$x = \text{argmax}_{x' \in \mathcal{X}} p_\theta(x'|z) \tag{64}$$

or sampling each variable independently (because of Eq. 63). These two strategies give rise to mappings from $\mathcal{Z}$ to $\mathcal{X}$ which are either deterministic (in argmax sampling strategy case) or stochastic. The mapping

$$z \mapsto \text{argmax}_{x' \in \mathcal{X}} p_\theta(x'|z) \tag{65}$$

is usually thought of defining the data manifold learned by a VAE.

Our approach is different from the one proposed in [26] since the latent variable z is only passed on the first time step of the decoder RNN and the variables are independent. We believe that in doing so, we "weaken the decoder" as it is recommended in [26] and force the decoder to use information from latent variable z.

We discuss more precisely the parametrization we used for the conditional distribution $p_\theta(x|z)$ and the approximated posterior $q_\phi(z|x)$ in Sect. 9.4.

Data Preprocessing

We extracted all monophonic soprano parts from the J.S. Bach chorales dataset as given in the music21 [38] Python package. We chose to discretize time with sixteenth notes and used the *real name* of notes as an encoding. Following the approach described in Chap. 6, we use the melodico-rhythmic encoding of Sect. 3.1.3 by adding an extra symbol which encodes that a note is held and not replayed. Every chorale melody is then transposed in all possible keys provided the transposition lies within the original voice ranges. Our dataset is composed of all contiguous subsequences of length $\Delta t = 32$ and we use a latent variable space with 12 dimensions. Our observation space is thus composed of sequences $x = (x_1, \ldots, x_{32}) \in \mathcal{X}$ where each element of the sequence x_i can be chosen between $A = 53$ different tokens.

Experimental Results

We choose to regularize one dimension by using as a function $g(x) := g_1(x)$ the number of played notes in the sequence x (it is an integer which is explicitly given by the representation we use).

IMPLEMENTATION DETAILS

We report the specifications for the model used in Sect. 9.3. All RNNs are implemented as 2-layer stacked LSTMs [69, 105] with 512 units per layer and dropout between layers; we do not find necessary to use more recent regularizations when specifying the non stochastic part model like Zoneout [83] or Recurrent batch normalization [36]. We choose as the probability distribution r_1 on the partial gradient norm a normal distribution with parameters $\mathcal{N}(2, 0.1)$.

We find out that the use of KL annealing was crucial in the training procedure. In order not to let the geodesic latent space regularization to be too important at the early stages of training, we also introduce an annealing coefficient for this regularization. This means we are maximizing the following regularized ELBO

$$\mathbf{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z) + \beta\, \mathcal{R}_{\text{geo}}(z; \{g_k\}, \theta)\right] - \beta\, D_{KL}(q_\phi(z|x)||p(z)) \quad (66)$$

with β slowly varying from 0 to 1. We also observe the necessity of early stopping which prevents from overfitting.

Structure of the latent space

Adding this regularization directly influences how the embedding into the latent space is performed by the VAE. We experimentally check that an increase Δz_1 in the first coordinate of the latent space variable $z = (z_1, \ldots, z_{\dim \mathcal{Z}})$ leads to an increase of

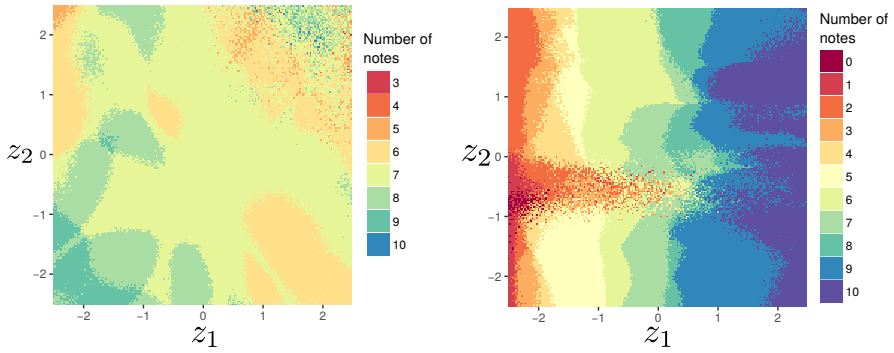
$$g_{\mathcal{Z}} := z \mapsto g(\text{argmax}(p_\theta(x|z))). \quad (67)$$

The (non-differentiable) function (Eq. 67) is in fact the real quantity of interest, even if it is the differentiable function G_1 (Eq. 72) which is involved in the geodesic latent space regularization (Eq. 59). In order to visualize the high-dimensional function $g_{\mathcal{Z}}$, we plot it on a 2-D plane containing the regularized dimension. In the remaining of this article, we always consider the plane

$$P_{z_1,z_2} = \{(z_1, z_2, 0, \ldots, 0), z_1, z_2 \in \mathbf{R}\} \quad (68)$$

Fig. 47 shows plots of the $g_{\mathcal{Z}}$ function restricted to the 2-D plane P_{z_1,z_2}. The case where no geodesic latent space regularization is applied is visible in Fig. 47a while the case where the regularization is applied on one latent space dimension is shown in Fig. 47b. There is a clear distinction between both cases: when the regularization is applied, the function $g_{\mathcal{Z}}$ is an increasing function on each horizontal line while there is no predictable pattern or structure in the non-regularized case.

In order to see if the geodesic latent space regularization has only effects on the regularized quantity (given by $g_{\mathcal{Z}}$) or also affects other



(a) Without geodesic latent space regularization

(b) With geodesic latent space regularization

Figure 47: Plot of a 2-D plane in the latent variable space $\mathcal{Z}$. The x-axis corresponds to the regularized dimension.

(non regularized) attribute functions, we plot as in Fig. 47 these attribute functions (considered as real functions on $\mathcal{Z}$ as in Eq. 67). Figure 48 show plots of different attribute functions such as the highest and lowest MIDI pitch of the sequence and the presence of sharps or flats. We remark that adding the latent space regularization tends to decorrelate the regularized quantities from the non-regularized ones.

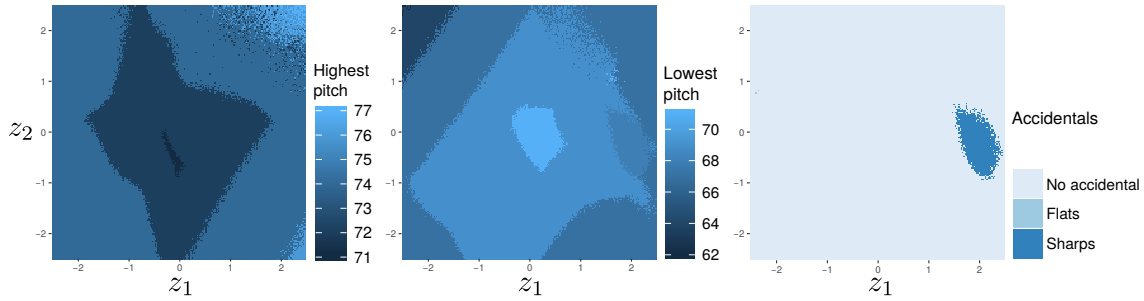
Generating Variations by moving in the latent space

Reducing correlations between features so that each feature best accounts for only one high-level attribute is often a desired property [32] since it can lead to better generalization and non-redundant representations. This kind of "orthogonal features" is in particular highly suitable for interactive music generation. Indeed, from a musical point of view, it is interesting to be able to generate variations of a given sequence with more notes for instance while the other attributes of the sequence remain unchanged.

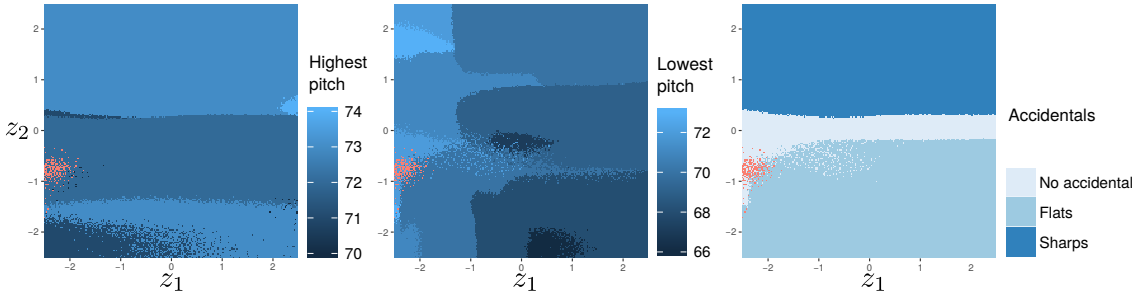
The problem of sampling sequences with a fixed number of notes with the correct data distribution has been, for example, addressed in [121] in the context of sequence generation with Markov Chains. In our present case, we have the possibility to progressively add notes to an existing sequence by simply moving with equal steps in the regularized dimension. We show in Fig. 49 how moving only in the regularized dimension of the latent space gives rise to variations of an initial starting sequence in an *intended* way.

Effect on the aggregated distribution and validation accuracy

A natural question which arises is: does adding a geodesic regularization on the latent space deteriorates the effect of the Kullback-Leibler regularization or the reconstruction accuracy? The possibility to sam-



(a) Without geodesic latent space regularization



(b) With geodesic latent space regularization

Figure 48: Plot of a 2-D plane in the latent variable space $\mathcal{Z}$. The x-axis corresponds to the regularized dimension. Different non-regularized quantities of the decoded sequences are displayed: the highest pitch, the lowest pitch and if the sequence contains no accidental, sharps or flats.

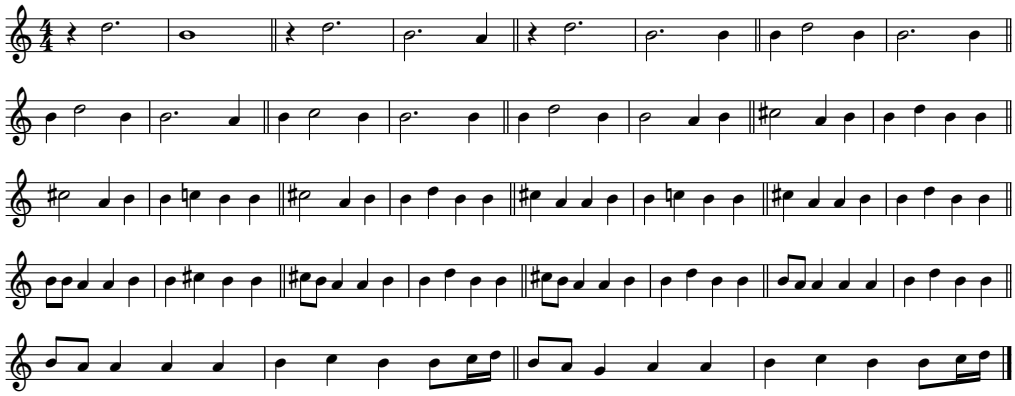


Figure 49: Image of straight line in the data manifold obtained by starting from a random z and then only increasing its first (regularized) coordinate z_1. The argmax decoding procedure (Eq. 64) was used. All generated sequences are two-bar long and separated by double bar lines. This generates variations of the initial motif by adding more notes.

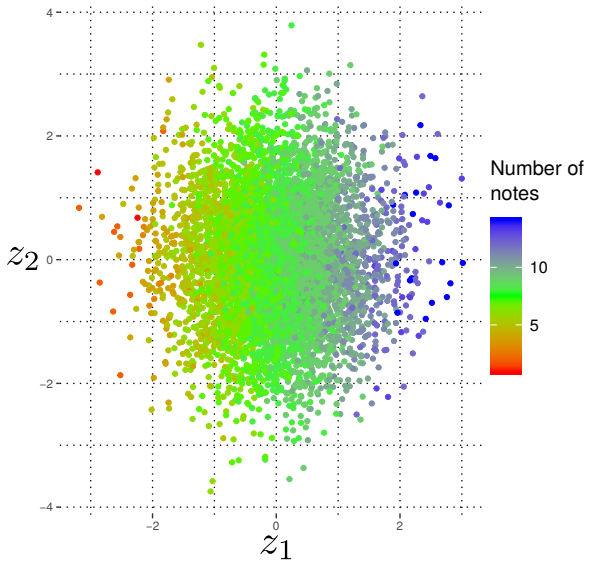


Figure 50: Plot of the aggregated distribution projected on a 2-D plane in the latent space which contains the regularized dimension as its x-axis. Each point is attributed a color depending on the number of notes contained in the decoded sequence.

ple from the data distribution by simply drawing a latent variable $z \sim p(z)$ from the prior distribution and then drawing $x \sim p_\theta(x|z)$ from the conditional distribution indeed constitutes one of the great advantage of the VAE architecture.

We check this by looking at the *aggregated distribution* defined by

$$q_\phi(z) := \int_x q_\phi(z|x)p_d(x)dx, \tag{69}$$

where $p_d(x)$ denotes the data distribution. In an ideal setting, where $q_\phi(z|x)$ perfectly matches the posterior $p_\theta(z|x)$, the aggregated distribution $q_\phi(z)$ should match the prior $p(z)$. We experimentally verify this by plotting the aggregated distribution projected on a 2-D plane in Fig. 50. By assigning colors depending on the regularized quantity, we notice that even if the global aggregated distribution is normally distributed and approach the prior, the aggregated distribution of each cluster of sequences (clustered depending on the number of notes they contain) is not, and depends on the regularized dimension.

We report a slight drop (1%) of the reconstruction accuracy when adding the geodesic latent space regularization. The fact that adding a regularization term reduces the reconstruction accuracy has also been noted in [84] where they nonetheless report a better visual quality for their regularized model.

The geodesic latent space regularization thus permits to obtain more meaningful posterior distributions while maintaining the possibility to sample using the prior distribution at the price of a small drop in the reconstruction accuracy. We believe that devising adap-

tive geodesic latent space regularizations could be a way to prevent this slight deterioration in the model's performance and provide us with the best of both worlds. Having the possibility to navigate in the latent space seems an important and desired feature for generative models in creative applications.

CHOICE OF THE REGULARIZATION PARAMETERS

We highlight in this section the importance of the choice of the regularization distributions r_k on the partial gradients (Eq. 59). Typical choices for r_k seem to be univariate normal distributions $\mathcal{N}(\mu, \sigma^2)$ of mean $\mu > 0$ and standard deviation σ.

In practice, this distribution's mean value must be adapted to the values of the attribute functions g_k over the dataset. Its variance must be small enough so that the distribution effectively regularizes the model but high enough so that the model's performance is not drastically reduced and training still possible. Fig. 51 displays the same figure as in Fig. 47b, but with a regularization distribution r_1 being a normal distribution $\mathcal{N}(5, 1)$. We see that imposing too big an increase to fit in the prior's range can cause unsuspected effects: we see here a clear separation between two phases.

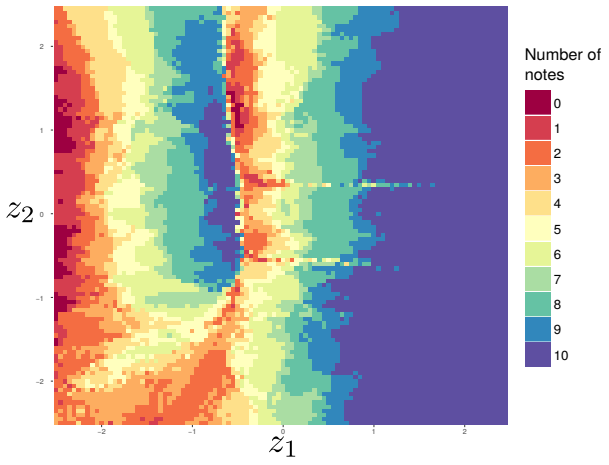


Figure 51: Same plot as in Fig. 47b but with $r_1 = \mathcal{N}(5, 1)$.

What is the best choice for the r_k distributions depending on the values of the g_k functions over the dataset is an open question we would like to address in future works.

DISCUSSION AND CONCLUSION

We introduced a new regularization function on the latent space of a VAE. This geodesic latent space regularization aims at binding a displacement in some directions of the latent space to a qualitative change of the attributes of the decoded sequences. We demonstrated

its efficiency on a music generation task by providing a way to generate variations of a given melody in a prescribed way.

Our experiments shows that adding this regularization allows interpolations in the latent space to be more meaningful, gives a notion of geodesic distance to the latent space and provides latent space variables with less correlation between its regularized and non-regularized coordinates.

Future work will aim at generalizing this regularization to variational autoencoders with multiple stochastic layers. It could indeed be a way to tackle the issue of inactive units in the lower stochastic layers as noted in [26, 97], by forcing these lower layers to account for high-level attributes.

Our regularization scheme is general and can be applied to the most recent generalizations of variational autoencoders which introduce generative adversarial training in order to obtain better approximations of the posterior distributions [75, 98] or in order to obtain a better similarity metric [86].

We believe that applying this regularization to conditional VAEs [141, 162] opens up new ways to devise interactive applications in a variety of content generation tasks.

Part IV

APPENDIX

A

EXAMPLES OF GENERATED MUSIC

This appendix presents generations using the DeepBach model. Section A.1 displays non interactive generations. These samples are shown in order to assess that the DeepBach model has good generative performance: it does not overfit the training data (or plagiarize) since it is able to generate original reharmonizations of existing melodies. We also discuss about the quality of the generated pieces from a musical point of view by analyzing its flaws and achievements. Section A.2 then shows many pieces produced using the interactive editor (Sect. 6.4). The aim is to show that it is possible to generate contrasting pieces in a simple way[1].

NON INTERACTIVE GENERATIONS

We now provide three complete chorale reharmonizations composed by DeepBach. One is a reharmonization of a chorale melody used by Bach (see Fig. 6) while the other two are different reharmonizations of the traditional hymn "God Save the Queen"(see Fig. 53 and 54).

These examples demonstrate the ability of DeepBach to learn and generate characteristic elements of J.S. Bach chorales while reharmonizing. To make our claim clearer, we highlighted particular aspects on the music sheets using three different colors:

- in *green*, we indicated:
  - characteristic melodic movements:
    - Fig 52 bars 1, 3, 6, 7, 9, 14
    - Fig 53 bars 13-14
    - Fig 54 bars 5, 13-14
  - good voicings and voice leading:
    - Fig 52 bars 2, 11
    - Fig 53 bars 2, 9
    - Fig 54 bars 2, 4, 13
  - characteristic suspensions[2] and passing tones:
    - Fig 52 bars 4, 8, 8-9, 14
    - Fig 53 bars 4, 13

1 Audio and videos of some of these pieces played live can be watched on https://sites.google.com/site/deepbachexamples/

2 For explanations for technical musical terms see https://en.wikipedia.org/wiki/Nonchord_tone

        * Fig 54 bar 4

- in *blue*:
  - musically interesting ideas:
    * Fig 52:
      · Starting on a dominant bar 1
      · Chromatic neighboring tone on the second degree bars 1, 13
      · Two different harmonizations between bars 1 and 8
      · Harmonization in A major bars 5-6
      · Bass in eighth notes bars 11-13
      · Cadence bar 12
    * Fig 53:
      · Starting in E minor bar 1
      · Harmonization in G minor bar 5
      · Chromatic line bars 11-12
      · Proximity between F and F# bars 11-12
    * Fig 54:
      · Dominant of the sixth degree
      · Minorization after a half cadence bars 6-7
      · Considering G at soprano to be an escape tone

- in *red*:
  - parallel fifths and octaves indicated by lines
  - mistakes:
    * Fig 52:
      · D missing bar 4
      · C should resolve to B bar 9
    * Fig 53:
      · E should be removed in order to prevent parallel fifths bar 1
      · Seventh chord cannot be played without preparation bar 9
      · Repetition in eighth notes bar 11
    * Fig 54:
      · Starting on an inverted chord of the first degree bar 1

- Strange resolution for 9-8 suspension bar 10
- Melodic movement is not Bach-like bar 11 (but it is imposed by the user and not generated by DeepBach)

Despite some compositional errors like parallel octaves, the musical analysis reveals that the DeepBach compositions reproduce typical Bach-like patterns, from characteristic cadences to the expressive use of nonchord tones. Furthermore, our model is able to propose varied and contrasting ideas when reharmonizing the same melody as can be seen by comparing the two versions of "God Save the Queen".

INTERACTIVE GENERATIONS

We now report some DeepBach generations made using the interactive editor described in Sect. 6.4. These pieces were first played during a concert called "Partnership" on May 23, 2017 which premiered "new musical works composed by or co-composed with systems created using different machine learning methods". The pieces we provided were meant to be reharmonizations of automatically-generated folk songs by the folk-rnn system (Sect. 4.1.1.1). A more complete presentation of all the musical pieces can be found in [144]. Since the editor allows to change every note, we wanted the generations to look like compositions typical of the style learned by DeepBach. For this, we restricted ourselves to only regenerate whole measures using the music editor. We took the freedom to adjust the timing of the given melodies, e.g., doubling the duration of the notes, adapting the rhythm for pieces in 6/8, or transposing them so that they fit within the soprano voice range.

Even when a folk-rnn tune features some unusual or rare melodic motions (compared to most of the Lutheran hymns used in J.S. Bach chorale harmonisations), DeepBach is still able to produce a fluid chorale texture and provides new insights and harmonic contours to the original melody. It is also worth noting how identical passages of the soprano part are harmonised differently.

Borrowing musical material from one source to adapt it in a particular language is a common technique. It is an important aspect of the music creation and has been practiced throughout ages, from J.S. Bach (Sect. 2.3.4) to Stravinsky or Messiaen [11].

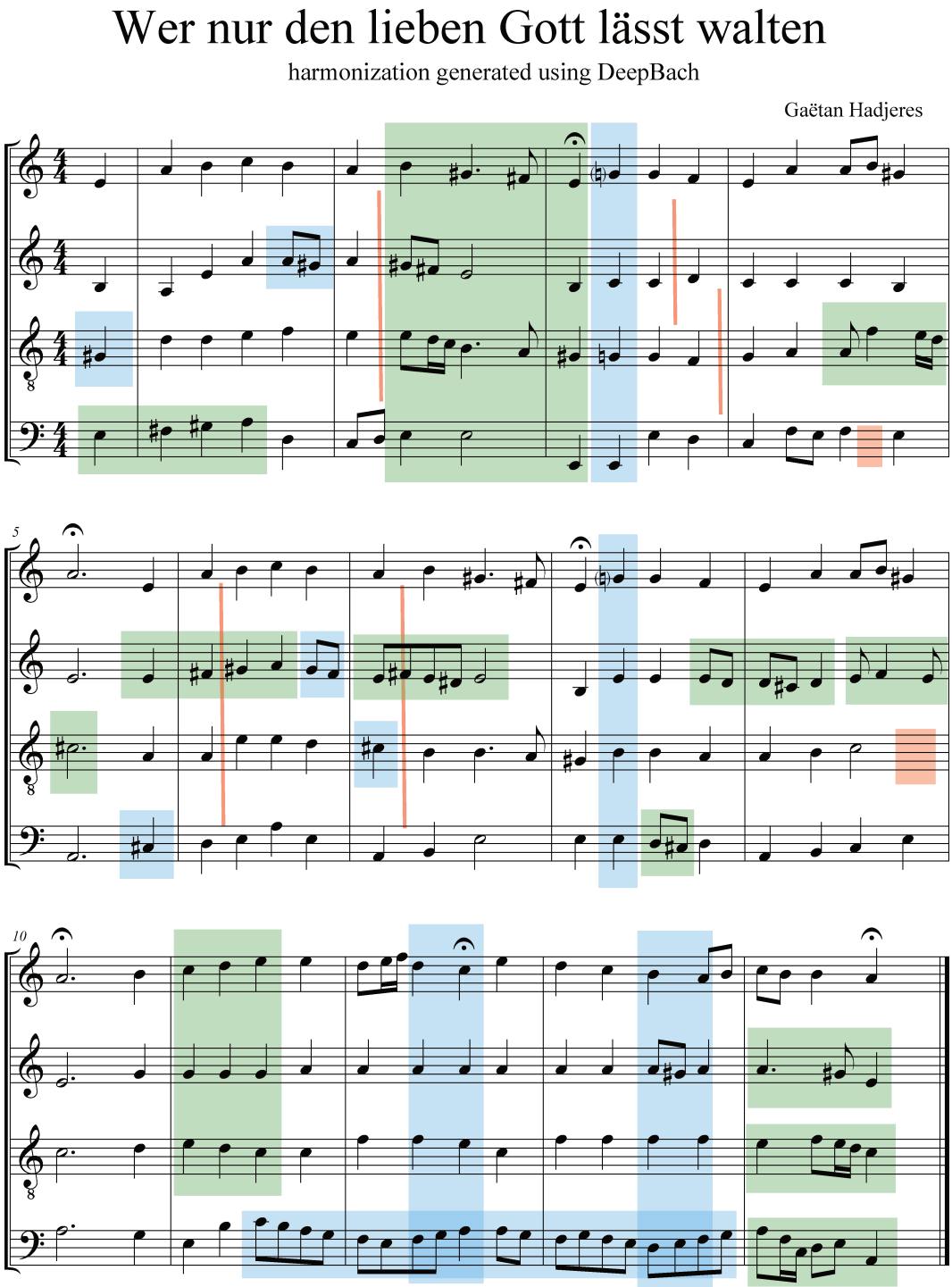


Figure 52: Reharmonization of "Wer nur den lieben Gott läßt walten" by DeepBach. See Sect. A.1 for comments on the annotations.

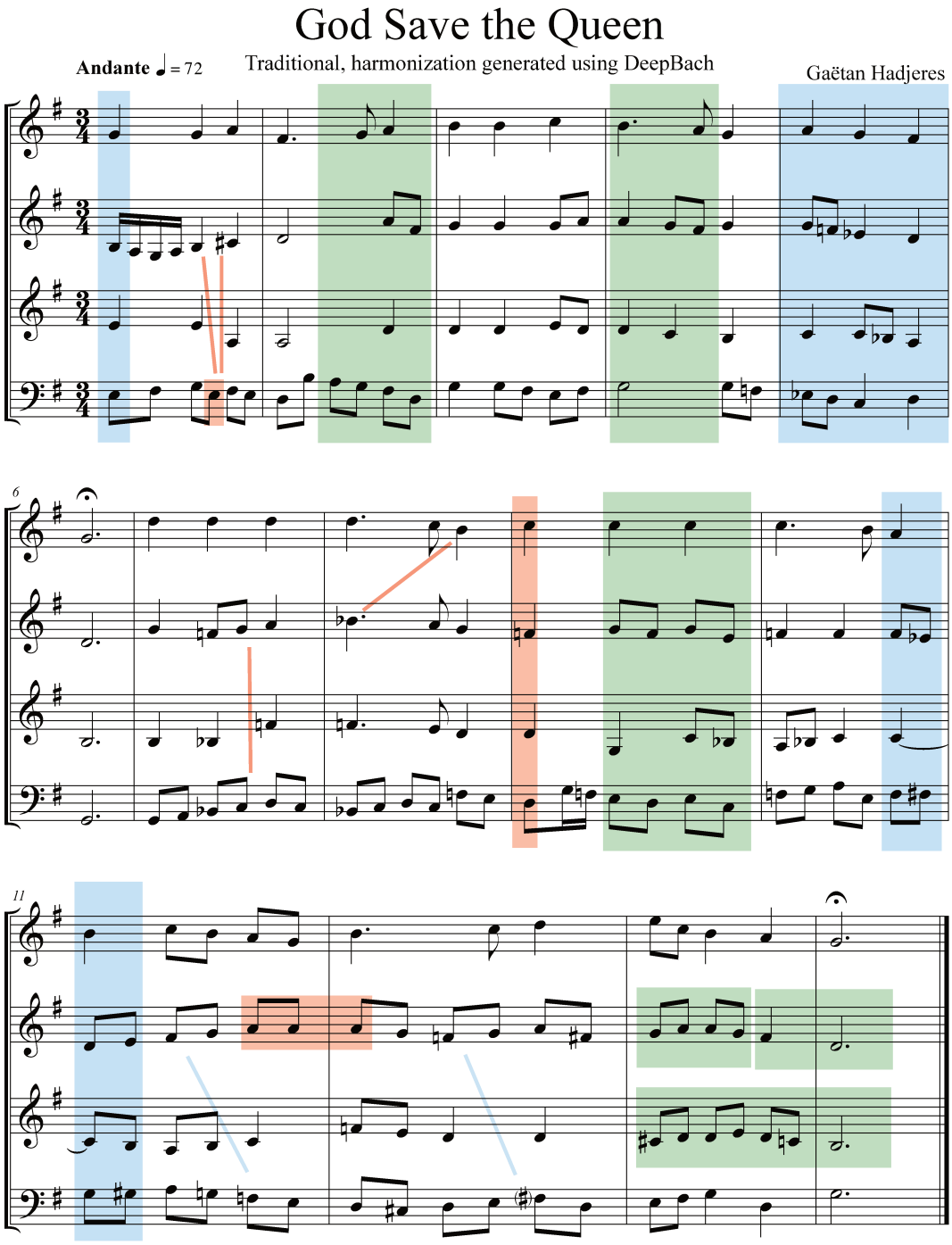


Figure 53: A reharmonization of "God Save the Queen" by DeepBach. See Sect. A.1 for comments on the annotations.

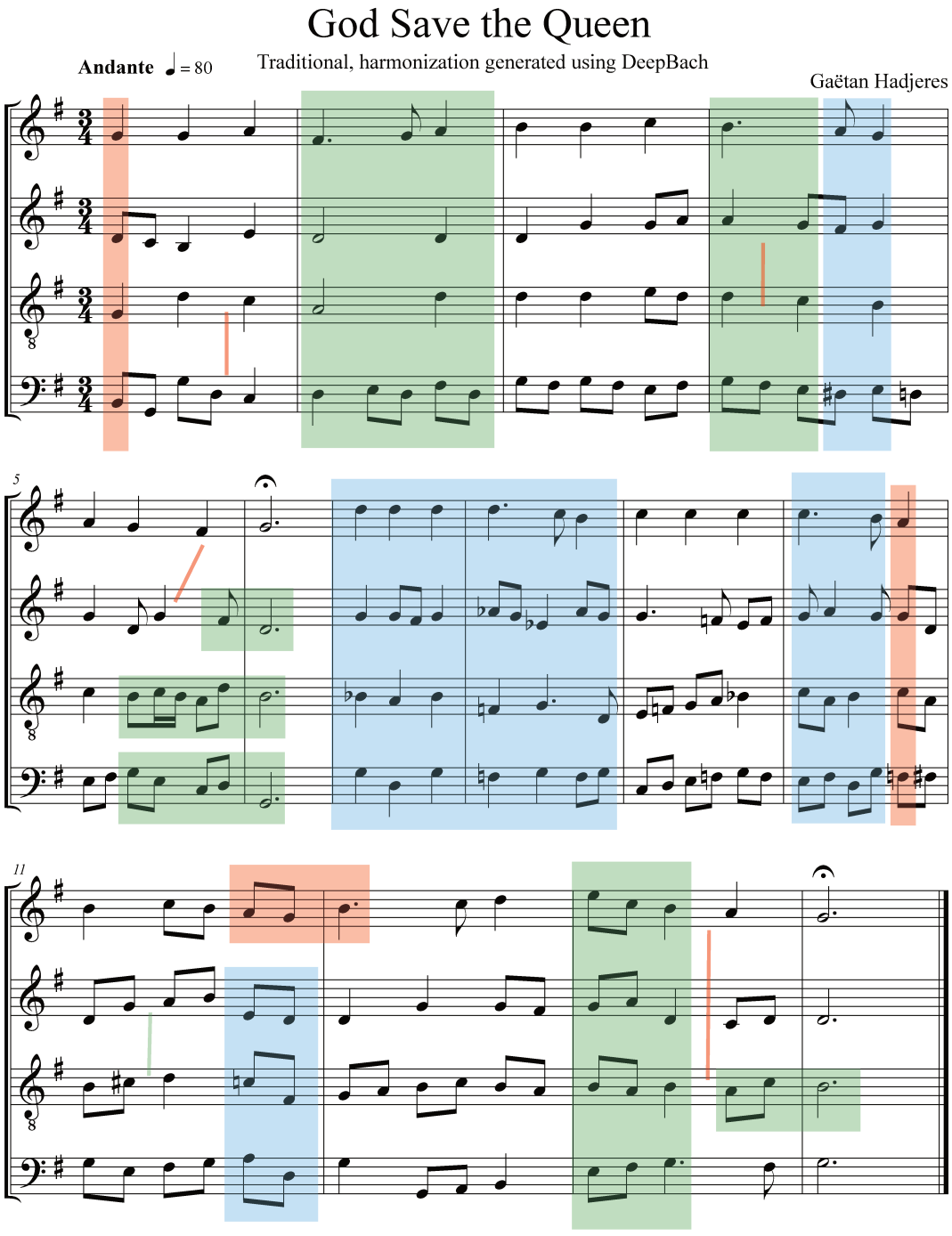
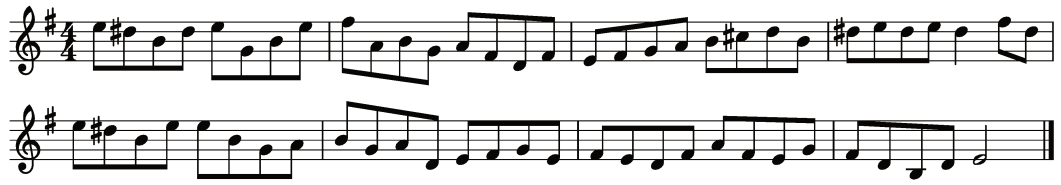
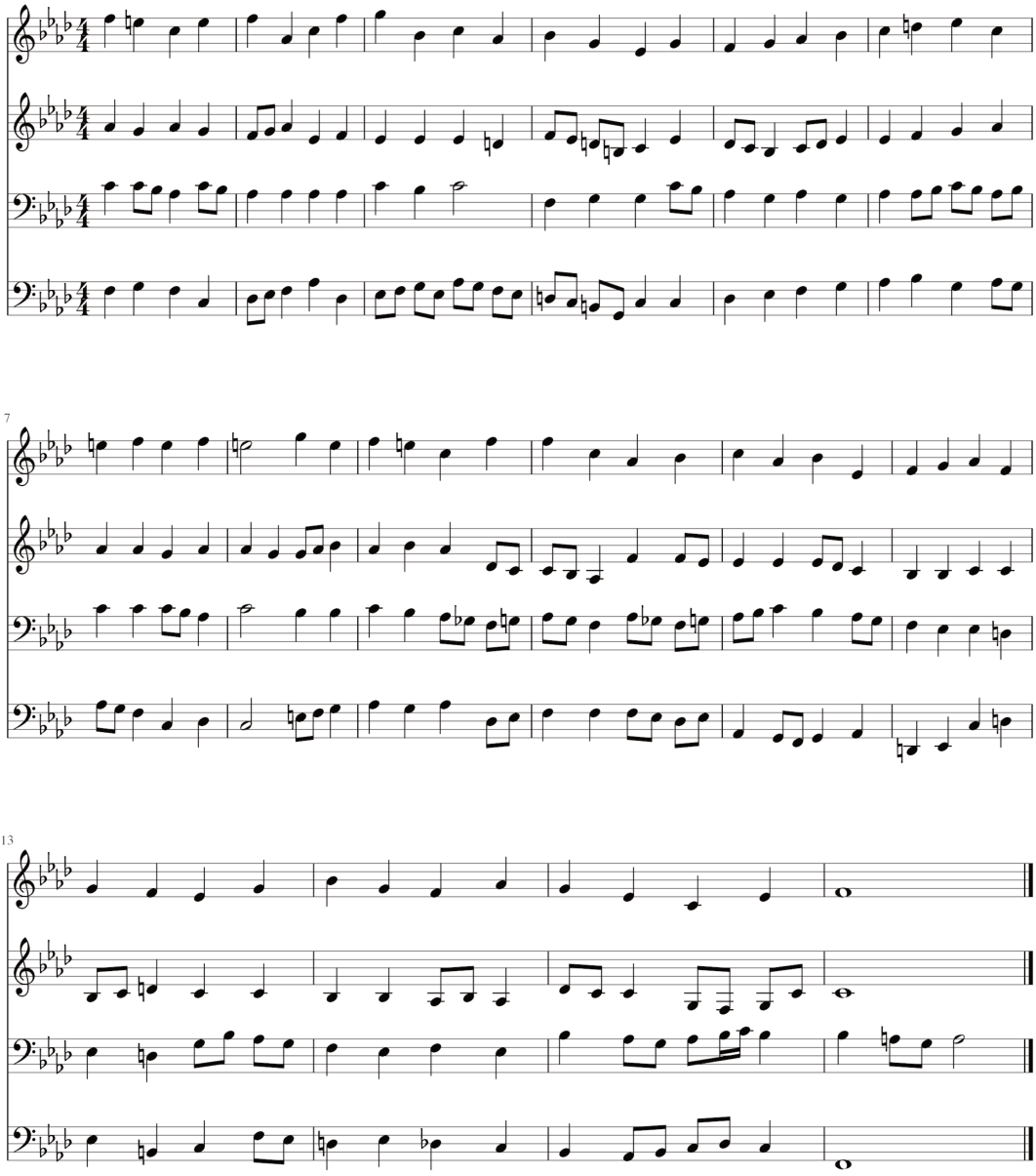


Figure 54: A second reharmonization of "God Save the Queen" by DeepBach. See Sect. A.1 for comments on the annotations.

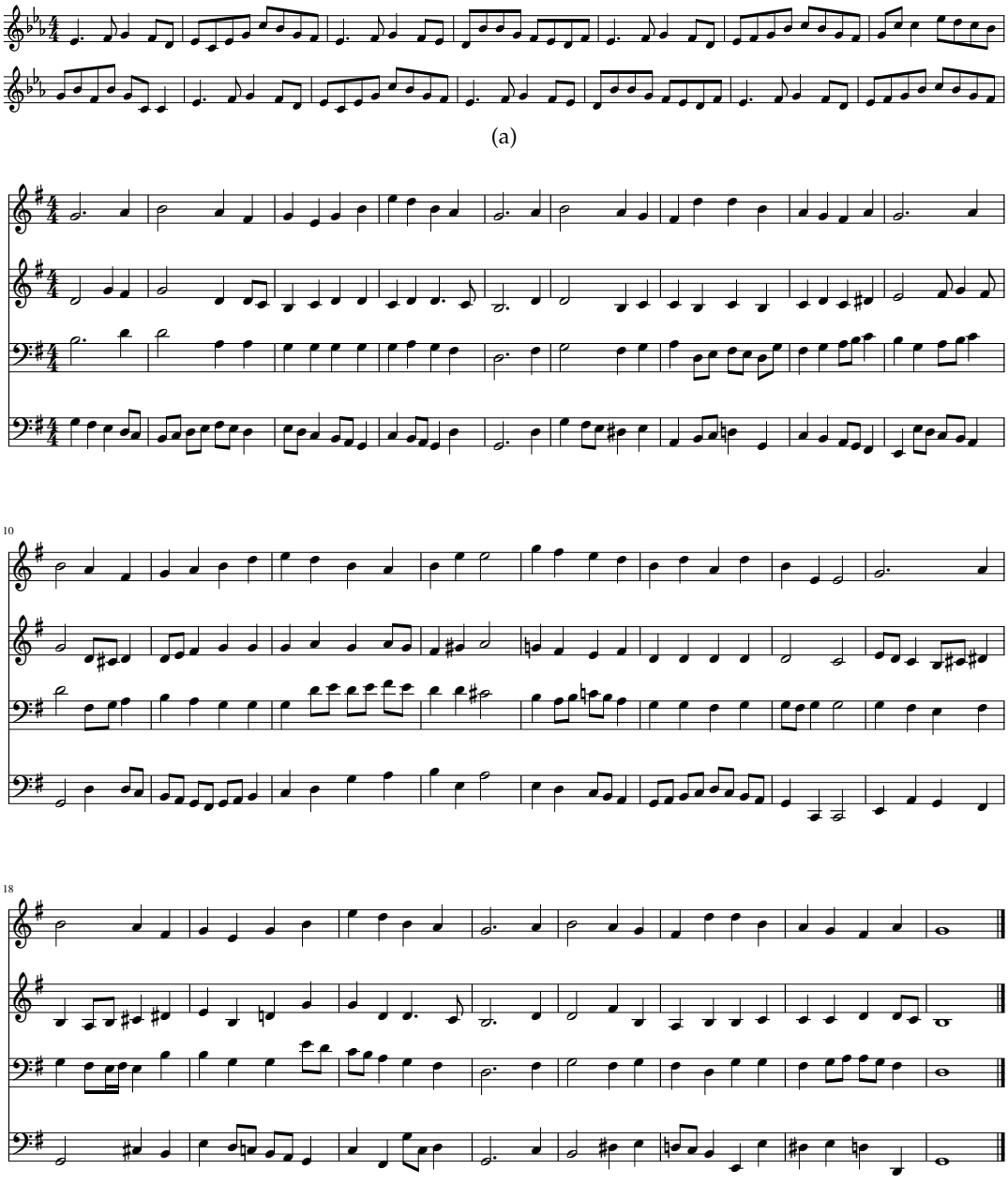


(a)



(b)

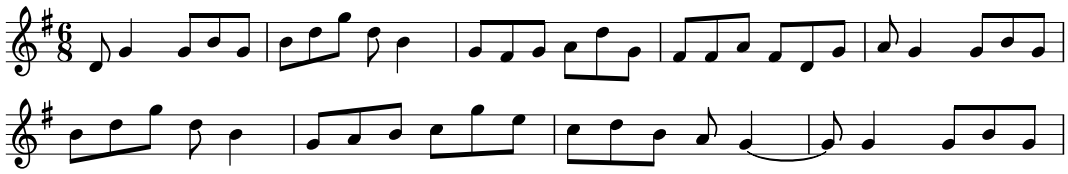
Figure 55: Generated folk-RNN tune #633 (a) and its reharmonization by DeepBach (b).



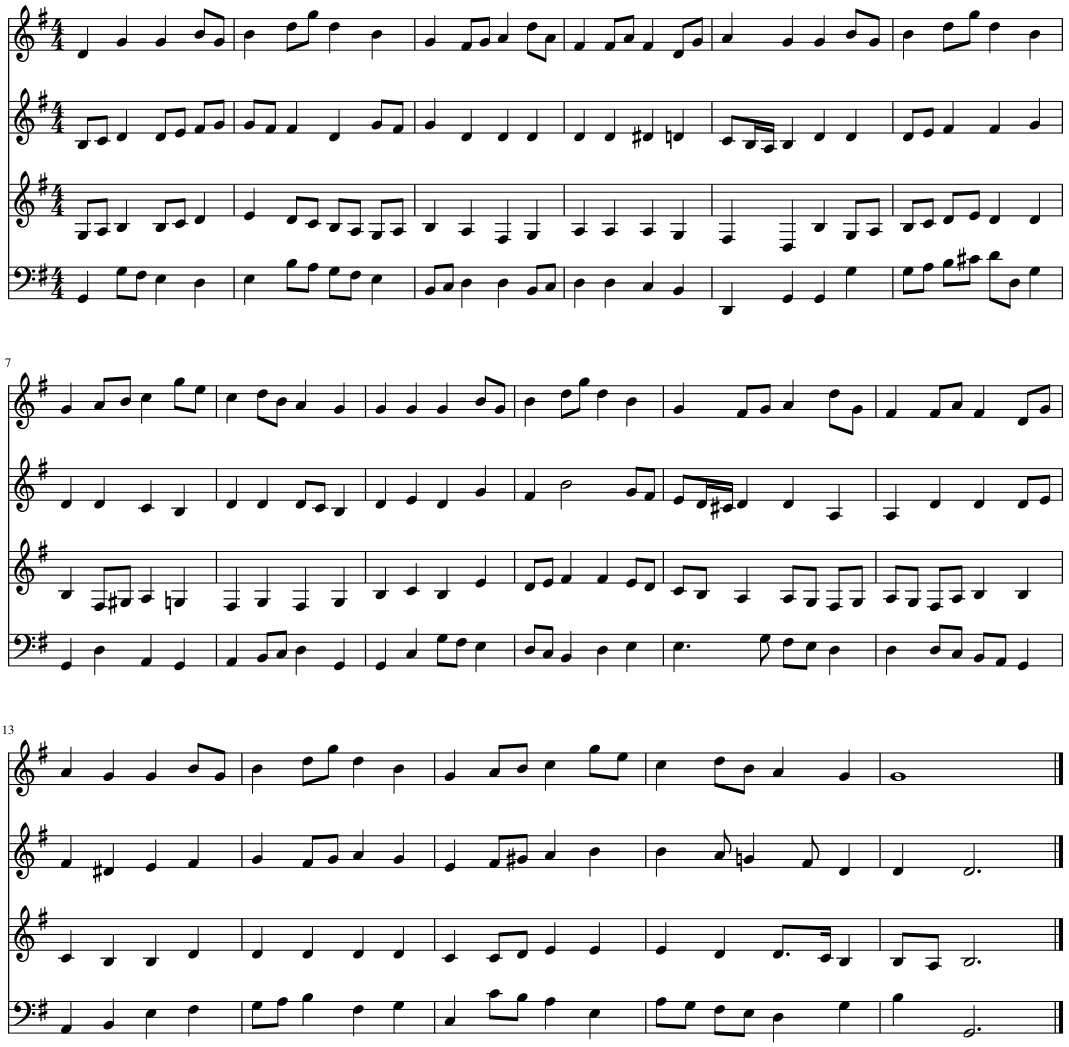
(a)

(b)

Figure 56: Generated folk-RNN tune #7153 (a) and its reharmonization by DeepBach (b).

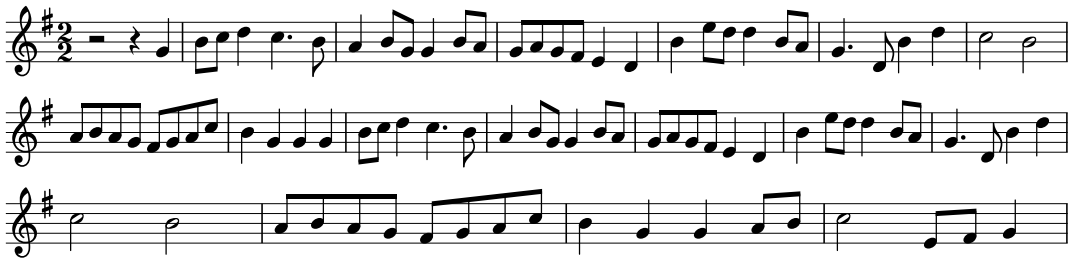


(a)

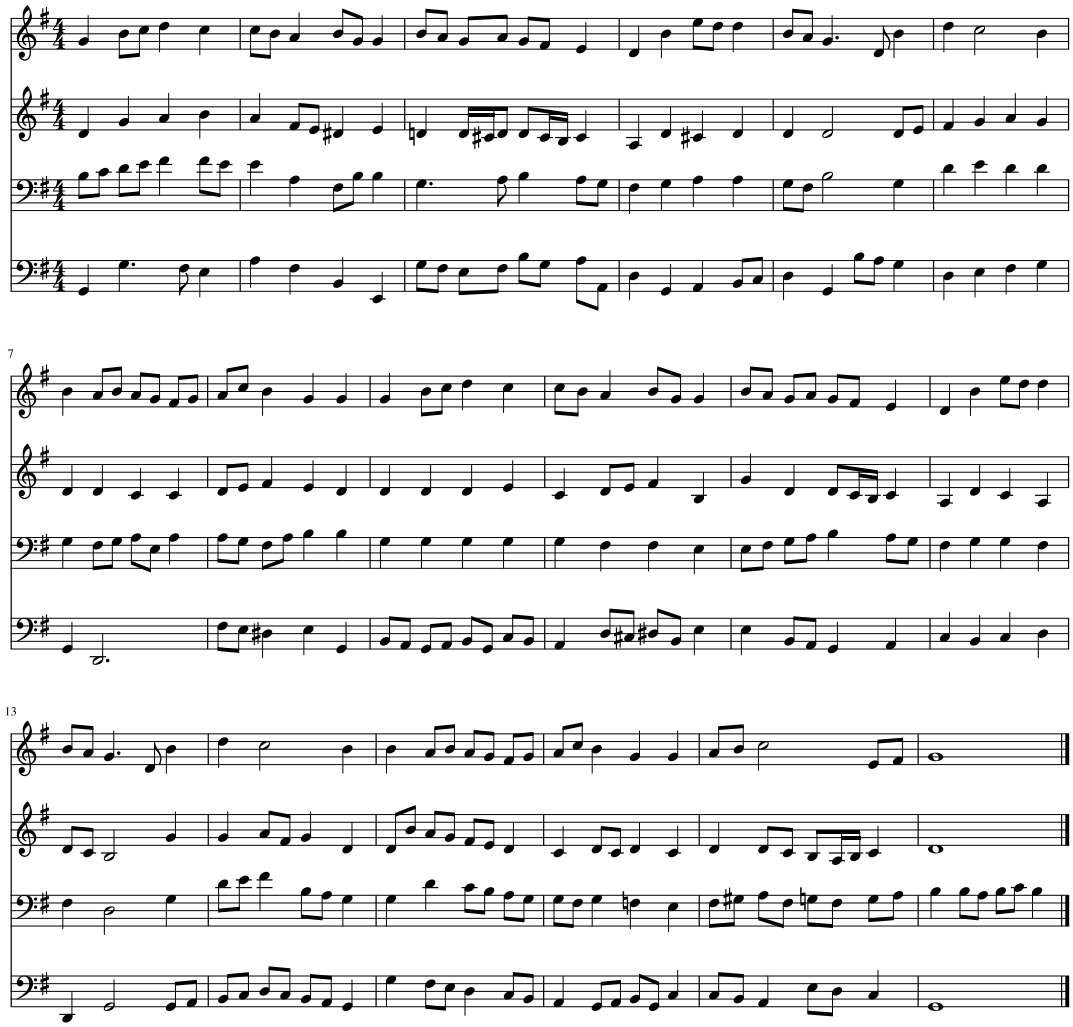


(b)

Figure 57: Generated folk-RNN tune named "The Glas Herry Comment" (a) and its reharmonization by DeepBach (b).



(a)



(b)

Figure 58: Generated folk-RNN tune named "The Deep Pint" (a) and its reharmonization by DeepBach (b).

B

RÉSUMÉ DE LA THÈSE

Cette partie est un résumé de cette thèse en français. L'objectif est de décrire de manière concise mais détaillée quelles ont été les motivations à ce travail et quelles sont les solutions proposées dans ce document. J'espère que ce court appendice donnera une idée claire des contributions de cette thèse et encouragera le lecteur intéressé à parcourir l'intégralité du manuscrit.

Les sources des programmes présentés dans cette thèse sont disponibles librement.

INTRODUCTION

Les avancées récentes dans le domaine de l'apprentissage profond (ou *deep learning*) permettent maintenant la création de modèles génératifs extrêmement performants. Avec les techniques actuelles, il est par exemple possible de générer une description textuelle à partir d'une image, une image correspondant à une description textuelle ainsi que de générer de générer du contenu audio à partir d'un texte. Si ces quelques exemples tirés de la littérature laissent entrevoir de nombreuses applications, les techniques utilisées ne s'appliquent pas aisément à la génération de partitions musicales (que nous appelleront également *musique symbolique* par opposition à la génération de musique dans le domaine audio).

En effet, la musique symbolique possède sa propre structure, distincte des autres types de données cités précédemment (audio, texte, images). Ainsi, la création de modèles génératifs pour la musique symbolique soulève de nouveaux problèmes en apprentissage profond et nécessite une approche spécifique.

Mais avant de poursuivre, interrogeons nous sur l'intérêt que peut avoir un algorithme générant automatiquement des partitions. Quel peut être son intérêt et pourquoi vouloir abandonner nos capacités artistiques au profit d'algorithmes ?

La composition musicale est en effet considérée comme un processus artistique dans lequel une intelligence artificielle n'aurait aucun rôle à jouer. Une oeuvre d'art est avant tout, à mon sens, la communication d'un processus créatif et d'une expérience humains. Détruire ce lien ténu entre l'artiste et son public n'est donc pas désirable. Si l'on peut apprécier la qualité d'une pièce générée par ordinateur, celle-ci sera difficilement acceptée telle quelle (c'est-à-dire sans la compréhension du processus créatif sous-jacent) en tant qu'oeuvre d'art.

Néanmoins, les systèmes de génération automatique de partitions de musique récemment proposés peuvent être perçus comme des boîtes noires capables de produire des partitions au kilomètre. Hormis lors de la conception du modèle génératif, aucune intervention humaine n'est présente. Ces systèmes, une fois entraînés, constituent "l'oeuvre d'art" et ne requièrent pas la présence d'une tierce personne.

Ces critiques ont déjà été formulées par le passé et certaines solutions ont été proposées, mais en dehors du contexte de l'apprentissage profond. Ces modèles permettent à utilisateur tiers (différent du concepteur du système) d'avoir une valeur ajoutée en composant à l'aide de ces outils. Cependant, ces systèmes, sont plus rigides et moins compétents que les modèles génératifs profonds.

Serait-il possible d'obtenir le meilleur des deux mondes ? Pouvons-nous concevoir des systèmes génératifs puissants remettant l'humain au centre de la composition ?

Ce sont ces problématiques que nous traiterons dans ce manuscrit. Plus précisément, nous chercherons à construire des modèles génératifs expressifs pouvant être utilisés à des fins artistiques par des utilisateurs extérieurs. Le thème de cette thèse se porte donc sur *les modèles génératifs profonds appliqués à la génération interactive de musique symbolique*, c'est-à-dire la conception de modèles génératifs avec lesquels un utilisateur peut interagir durant la composition. Cette thématique à l'intersection entre plusieurs domaine n'a jusqu'à lors jamais été énoncée de la sorte et j'espère que cette thèse ne sera pas la dernière sur ce sujet.

Je suis en effet convaincu qu'il est essentiel pour ces modèles génératifs appliqués à la musique de proposer une interaction avec un utilisateur externe. Cela permet de relier ces problèmes théoriques à des problématiques réelles. Ainsi, concevoir de puissants modèles génératifs pour la musique symbolique n'est pas suffisant, mais il faut également repenser l'interaction entre l'utilisateur et la machine. De cela découle une pléthore de problèmes tantôt concrets, tantôt théoriques. En effet, le potentiel de ces nouveaux outils de composition ne peut s'exprimer pleinement que par le biais d'interfaces utilisateur adaptées. Cette intrication entre les modèles génératifs d'une part et leur utilisation d'autre part pose de nouvelles questions :

- Quelles fonctionnalités sont nécessaires au cours de la composition assistée par ordinateur ?
- Peut-on améliorer la créativité ou la productivité des utilisateurs ?
- Sous quelle forme se présente l'interaction entre l'utilisateur et la machine ? Quel type d'interface utilisateur cette interaction implique-t-elle ?

- Inversement, les innovations dans le domaine des modèles génératifs profonds peuvent-elles suggérer de nouveaux processus compositionnels ?
- Comment concevoir des modèles génératifs expressifs et flexibles pouvant satisfaire des contraintes à la fois techniques et musicales ?

À mon sens, des motivations artistiques peuvent motiver le développement de nouveaux modèles en intelligence artificielle et vice-versa. Ce lien entre les deux domaines que sont l'intelligence artificielle et la musique est, selon moi, fructueux et à double sens. Utiliser ces modèles d'apprentissage profonds comme des outils artistiques permet de mieux cerner le comportement de ces modèles génératifs.

CONTRIBUTIONS

Cette thèse est composée de trois parties.

DONNÉES MUSICALES, CHALLENGES ET CRITIQUE DE L'ÉTAT DE L'ART

La première partie est une vue d'ensemble sur l'utilisation des modèles génératifs profonds appliqués à la musique symbolique. Le but n'est pas de constituer un état de l'art exhaustif mais plutôt d'identifier à travers quelques exemples représentatifs les différentes approches présentées ces dernières années. Cette partie est constituée de trois chapitres.

Données musicales symboliques

Le chapitre 2, *Données musicales symboliques*, présente de manière détaillée les différents formats utilisés pour noter, de manière symbolique, la musique. En effet, contrairement au texte ou aux images, il n'existe pas de standard pour les données musicales. Une partition (dans le système de musique occidentale) est avant tout une représentation graphique, et il semble évident que ce format n'est pas adapté à une approche algorithmique.

L'important dans le choix d'une notation est donc, comme dans tout système de communication, de trouver "le juste milieu" entre la précision de celle-ci, sa concision et son utilité. Une notation musicale sera particulièrement adaptée si ses utilisateurs, compositeurs et interprètes, sont capables de communiquer et comprendre leurs intentions musicales avec un minimum d'effort.

Les formats les plus courants sont présentés (notation occidentale sur cinq portées, leadsheets, MusicXML, notation ABC, MIDI) et leurs

avantages et désavantages discutés. Tout au long de cette thèse, nous feront la distinction entre le format sous lequel se présentent les données brutes (la notation musicale) et leur encodage (ou représentation) utilisé par l'ordinateur.

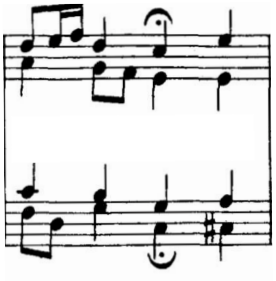
Les données musicales symboliques présentent des qualités distinctes des autres types données, ce qui fait qu'il n'est pas nécessairement aisé pour un public non musicien de comprendre ces spécificités. Ce chapitre se poursuit donc par une présentation succincte des aspects les plus importants à prendre en compte lors d'un traitement algorithmique de ces données. Les notions de rythme, de mélodie, d'harmonie, de style ainsi que les problématiques liées à la structure à petite ou grande échelle sont mises en avant. Il apparaît alors que le traitement algorithmique de la musique symbolique nécessite des modèles génératifs dédiés capables de capturer ces spécificités.

Ce chapitre se termine par la présentation des jeux de données de musique symbolique les plus importants. Une attention particulière est dévolue au dataset des harmonisations de chorals par Jean-Sébastien Bach. La particularité de ce corpus est son homogénéité et sa taille. En effet, celui-ci est composé d'environ 400 courtes pièces d'environ une minute et composées suivant les mêmes principes et pour la même formation (chorale à 4 voix). L'écriture des chorals de Bach est remarquable et est étudiée dans la plupart des cours d'écriture musicale depuis des siècles. Il est donc aisé pour un expert d'évaluer la qualité d'un choral généré par ordinateur, ce qui n'est pas nécessairement le cas avec d'autres styles de musique. Pour les raisons avancées ci-dessus, ces harmonisations de chorals constitueront le dataset de choix tout au long de cette thèse.

Les challenges de la génération de musique symbolique

Le chapitre 3, *Les challenges de la génération de musique symbolique*, traite l'ensemble des données musicales symboliques présentées au chapitre précédent sous l'angle de l'apprentissage automatique. En particulier, nous discutons des problèmes rencontrés lors de la conception de modèles génératifs profonds pour la musique symbolique.

La première observation est qu'une *représentation* adaptée doit être trouvée. Comme pour les notations musicales, la représentation ou l'encodage parfait n'existent pas. En revanche, il est important de connaître les différentes possibilités à disposition lors de la conception d'une représentation pour la musique symbolique ainsi que l'influence de ces choix sur le résultat de la génération. Les encodages usuellement utilisés pour les notes et le rythme sont présentés et leurs avantages respectifs analysés. De cette analyse découle la proposition d'un nouvel encodage nommé *encodage mélodico-rythmique* particulièrement adapté pour représenter des séquences monophoniques.



(a)

```
D5,__,E5,F5,D5,__,__,__,C5,__,__,__,E5
A4,__,__,__,G4,__,F4,__,E4,__,__,__,E4
C4,__,__,__,B3,__,__,__,G3,__,__,__,A3
F3,__,D3,__,G3,__,__,__,C2,__,__,__,C#2
```

(b)

Figure 59: Extrait d'un choral de J.S. Bach chorale à côté de son encodage mélodico-rythmique. Le symbole de continuation est noté "__".

Cette représentation peut également être utilisée pour des pièces écrites pour plusieurs instruments monophoniques. Le but de cet encodage est de représenter une séquence monophonique en une séquence de symboles. Contrairement à d'autres représentations existantes, il n'y a pas de dissociation entre le rythme et la mélodie (on pourrait représenter une mélodie comme une séquence de couples (note, durée)). Cet encodage mélodico-rythmique repose sur l'introduction d'un symbole additionnel "__" nommé *symbole de continuation* indiquant que la note précédente est tenue. Appliqué à des séquences monophoniques, cet encodage est non ambigu, sans perte d'information et compact.

Un exemple de cette encodage appliqué à un choral à quatre voix de J.S. Bach est montré à la Fig. 59.

Cet encodage se révélera particulièrement adapté pour échantillonner des séquences à l'aide de méthodes de Monte-Carlo par chaînes de Markov.

Le second challenge traité dans ce chapitre est le problème de l'évaluation de ces modèles génératifs pour la musique. En effet, comparer deux modèles génératifs sur une base objective est possible mais peu informative: une meilleure vraisemblance des données n'implique pas nécessairement la capacité de générer de la "meilleure musique". Il est cependant possible de mener des évaluations à l'aide de tests de perception. Mais là encore, les résultats obtenus sont difficilement généralisables et sont extrêmement dépendant de la manière dont l'expérience a été menée (l'expérience musicale des sujets étant un facteur important). Se pose également le problème du plagiat: quels critères intégrer afin de différencier une recopie d'un morceau existant d'une inspiration ou de la reprise d'un élément caractéristique ?

En revanche, l'évaluation des modèles génératifs interactifs couplés à des interfaces utilisateurs peut prendre un sens nouveau: un système génératif peut alors être considéré comme bon s'il permet aux utilisateurs de s'exprimer de manière intuitive, ergonomique et si la musique qu'il produit est écoutée. En d'autres mots, évaluer un sys-

tème génératif revient à évaluer si c'est un bon "produit" ou un bon outil. Cela répond d'une certaine manière au problème du plagiat : si un système génératif est performant et permet de créer de nouveaux contenus (considérés comme non plagiant) quand il est utilisé, cela est suffisant. Évaluer cette propriété au niveau du modèle génératif uniquement me semble manquer de sens.

Les modèles génératifs profonds pour la musique symbolique

Le dernier chapitre de la première partie présente certains modèles génératifs pour la musique récemment publiés. Un effort d'unification a été fait afin de percevoir les similitudes et différences entre ces approches. Nous verrons que la principale limitation de ces modèles est que l'interaction avec un utilisateur est faible voire inexistante, ce qui réduit grandement leur usage.

MODÉLISATION DE LA MUSIQUE POLYPHONIQUE

La seconde partie de cette thèse se concentre sur la modélisation de la musique polyphonique, et plus particulièrement sur la modélisation des chorals de J.S. Bach. La musique polyphonique a l'avantage d'être beaucoup plus "stricte" et plus "combinatoire" que la musique monophonique. En d'autres termes, il est beaucoup plus facile de repérer une "fausse note", même pour une personne non musicienne, dans le premier cas.

Cette partie est composée de deux chapitres. Le premier, *Familles exponentielles pour l'imitation du style et l'invention d'accords dans la musique polyphonique* , présente un modèle probabiliste pour la musique de style choral. Cette approche introduit une méthode de Monte-Carlo par chaînes de Markov pour échantillonner de la musique polyphonique. L'avantage de cette méthode est sa flexibilité et sa formulation théorique claire. En revanche, le modèle probabiliste utilisé manque de capacité et les pièces générées, si elles sont convaincantes, ne sont pas en mesure d'être confondues avec les pièces originales de J.S. Bach. Le second chapitre, *DeepBach: un modèle contrôlable pour la génération de chorals*, raffine et améliore cette approche. Le modèle probabiliste est remplacé par un réseau de neurones profond ce qui améliore grandement la qualité des pièces générées. Cette amélioration s'effectue au détriment de la formulation théorique plus claire. Un plug-in pour l'éditeur de partitions MuseScore a été développé afin rendre l'interaction avec un utilisateur possible. Cela permet d'introduire une part d'intelligence artificielle dans le processus compositionnel de manière naturelle et intuitive.

Familles exponentielles pour l'imitation du style et l'invention d'accords dans la musique polyphonique

Ce chapitre introduit un modèle probabiliste sur des séquences polyphoniques ayant une représentation matricielle similaire à celle présentée Fig. 59. L'idée est d'écrire la probabilité paramétrée par le paramètre θ d'une séquence s sous la forme

$$P(s|\theta) = \frac{e^{-E(s,\theta)}}{Z(\theta)}, \tag{70}$$

où l'*énergie* d'une séquence $E(s,\theta)$ est écrite :

$$E(s,\theta) := -\sum_{ab,ijk} \theta^{ab,ijk} f_{ab,ijk}(s). \tag{71}$$

Dans l'expression précédente,

$$Z(\theta) := \log(\sum_{s} e^{-E(s,\theta)})$$

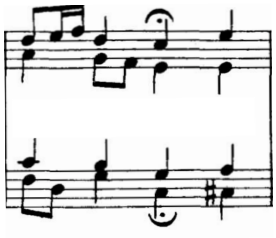
est la fonction de partition telle que la somme des probabilités soit égale à 1. Les statistiques $f_{ab,ijk}$ considérées ici représentent le nombre de co-occurrences de la note a à la voix i et de la note b à la voix j étant séparées de k pas de temps.

L'optimisation du paramètre θ s'effectue en maximisant la pseudo-log-vraisemblance des données d'entraînement, c'est-à-dire la maximisation de la log-vraisemblance des probabilités conditionnelles d'une note sachant les autres notes.

Le point intéressant dans ce modèle est que la probabilité d'une note donnée dépend des notes la précédent, comme dans la majorité des approches existantes où la génération s'effectue de gauche à droite, mais également des notes qui la suivent. Les contextes passé et futur sont donc utilisés.

La génération s'effectue à l'aide de l'algorithme de Metropolis-Hastings. C'est une méthode de Monte-Carlo avec chaînes de Markov qui modifie incrémentalement les échantillons. Dans les faits, et étant donné notre choix de $P(s|\theta)$, cet algorithme ne requiert que la connaissance des probabilités conditionnelles qui ont une expression particulièrement simple. La force de cette approche est qu'il est possible de modifier légèrement cet algorithme génératif afin de pouvoir contraindre certaines notes à avoir une valeur prédéfinie. Cela permet par exemple de pouvoir réharmoniser une mélodie donnée ou de commencer et terminer par un accord donné.

Ces deux éléments (prise en compte du contexte passé et futur ainsi que la génération non séquentielle et itérative) sont à mon sens plus proches des pratiques compostionnelles réelles. Il est en effet très rare qu'un compositeur compose de gauche à droite en une seule passe. Au contraire, certains éléments sont écrits, puis arrangés et agencés correctement.



(a)

```
D5,__,E5,F5,D5,__,__,__,C5,__,__,__,E5
A4,__,__,__,G4,__,F4,__,E4,__,__,__,E4
C4,__,__,__,B3,__,__,__,G3,__,__,__,A3
F3,__,D3,__,G3,__,__,__,C2,__,__,__,C#2
 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1
 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0
```

(b)

Figure 60: Extrait d'un choral de J.S. Bach chorale à côté de son encodage mélodico-rythmique. Le symbole de continuation est noté "__-". Les cinquième et sixième voix contiennent respectivement des indications métronomiques et la présence ou absence de points d'orgue.

DeepBach: un modèle contrôlable pour la génération de chorals

Ce chapitre apporte une solution aux lacunes du modèle précédent tout en en conservant ses avantages. La modélisation des données est plus précise et repose utilise l'encodage mélodico-harmonique. Les spécificités du corpus des chorales de J.S. Bach sont également prises en compte. En particulier, la présence des points d'orgue (qui dans les chorals de Bach indiquent une fin de phrase) est considérée et des informations rythmiques sont ajoutées. Cela se manifeste par l'introduction de voix supplémentaires dans la représentation mélodico-harmonique (voir Fig. 60).

L'amélioration essentielle provient du modèle probabiliste utilisé. Le constat est le suivant : dans le modèle exposé au chapitre précédent, seules les probabilités conditionnelles (la probabilité d'une note sachant toutes les autres notes) sont utilisées, que ce soit durant l'entraînement du modèle en maximisant la pseudo-log-vraisemblance ou durant l'échantillonnage à l'aide de l'algorithme de Metropolis-Hastings. Le modèle DeepBach approxime alors directement à l'aide de réseaux profonds ces probabilités conditionnelles et abandonne donc l'expression en forme close Eq. 70 de la probabilité globale d'une séquence. Cette modélisation, où seules les probabilités conditionnelles sont approximées sans que celles-ci soient obtenues à partir d'une probabilité globale sur toutes les variables, est un exemple de *réseau de dépendances*.

Afin de modéliser la probabilité conditionnelle d'une note sachant toutes les autres notes (son contexte), l'architecture retenue repose sur deux réseaux de neurones récurrents chargés de résumer respectivement les contextes passé et futur ainsi qu'un réseau de neurones à propagation avant pour résumer le contexte présent (les notes jouées simultanément). Une représentation graphique de cette architecture est visible Fig. 61.

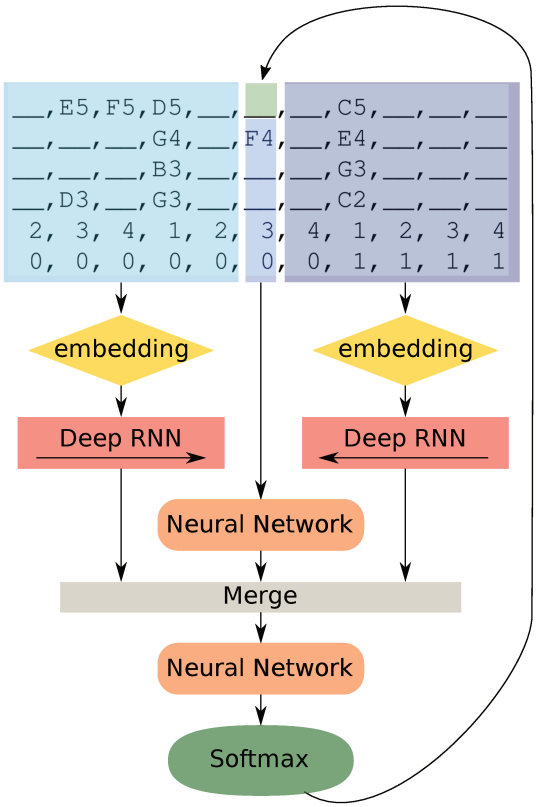


Figure 61: Représentation graphique de l'architecture utilisée dans le modèle DeepBach afin de modéliser les probabilités conditionnelles. L'objectif ici est de classifier correctement la note en vert sachant son contexte.

L'échantillonnage est alors possible en utilisant le pseudo-échantillonnage de Gibbs. Ce chapitre commente alors le choix de la représentation utilisée. En particulier, son adéquation avec cette méthode d'échantillonage est mise en avant.

Afin d'évaluer la qualité des générations produites par ce modèle, une expérience en ligne a été mise en place à laquelle plus de mille participants ayant des éducations musicales différentes ont pris part. L'objectif pour les participants était de déterminer si un extrait de choral était original, composé par J.S. Bach, ou généré en utilisant DeepBach. Les résultats sont très satisfaisant : environ la moitié des votes (incluant des musiciens professionnels) ont attribué des chorals générés par DeepBach à J.S. Bach.

Une interface utilisateur permettant d'utiliser DeepBach de manière aisée a été développée. Celle-ci se présente sous la forme d'un plug-in pour l'éditeur de partitions MuseScore. L'idée est de pouvoir générer une sélection quelconque à l'aide de DeepBach directement depuis cet éditeur de partitions. Ainsi, l'utilisateur bénéficie de toutes les possibilités de micro-édition disponibles dans cet éditeur, mais peut également interagir avec DeepBach afin de composer plus rapidement ou explorer rapidement des idées nouvelles. Un exemple de cette interface et de son intégration dans MuseScore est montrée Fig. 62.

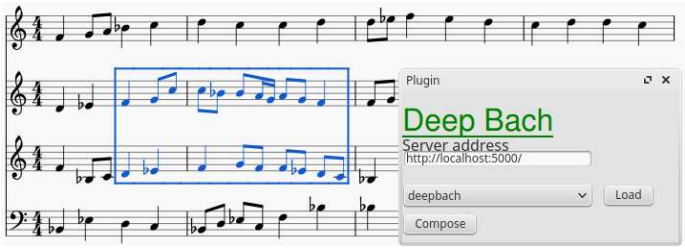


Figure 62: Interface du plug-in DeepBach pour Musescore.

Avec cette interface, une personne n'étant pas musicienne ou n'étant pas experte peut composer des chorals de bonne facture en très peu de temps. J'espère que cela permettra de rendre plus accessible et plus attractive cette musique à un public non initié. Il y a en effet un aspect ludique indéniable dans cette nouvelle manière de composer.

De nombreux exemples de chorals générés, de manière interactive ou non, sont présentés en appendice. Une analyse musicale a été faite pour certains d'entre eux afin de mettre en lumière les capacités et lacunes de ce modèle génératif.

TECHNIQUES NOUVELLES POUR LA GÉNÉRATION SÉQUENTIELLE

Cette dernière partie est composée de trois chapitres distincts. Chaque chapitre est consacré au développement d'une nouvelle tech-

nique motivée par et ayant des applications pour la génération de musique symbolique. Ces techniques sont dans les faits plus générales et s'appliquent à la génération de séquences quelconques. Dans toute cette partie, on considère les séquences musicales comme des séquences de symboles.

Le premier chapitre, *Distances de rang invariantes par transposition pour des séquences musicales*, introduit une distance entre séquences musicales ayant propriété d'être invariante par transposition. Cela est particulièrement intéressant puisque notre conception de la similarité est relative et ne dépend pas d'une hauteur de référence. Le second chapitre, *Anticipation-RNN: Génération interactive de musique sujette à des contraintes unaires*, développe une architecture efficace permettant de générer des séquences musicales satisfaisant un ensemble de contraintes unaires. L'intérêt est de pouvoir se passer des méthodes de Monte-Carlo par chaînes de Markov, plutôt lentes par nature, tout en conservant des possibilités de contrôles similaires à celles présentes dans l'échantillonnage du modèle DeepBach. Cette thèse se conclue par le chapitre *GLSR-VAE: Régularisation géodésique de l'espace latent pour les auto-encodeurs variationnels*. La motivation derrière cette contribution est de pouvoir structurer l'espace latent des auto-encodeurs variationnels afin que ceux-ci puissent être utilisés de manière interactive. En effet, dans l'espace latent d'un auto-encodeur variationnel classique, il n'existe pas de direction privilégiée et les axes des coordonnées ne possèdent pas de signification particulière. Le but de ce chapitre est une régularisation de l'objectif usuel visant à rendre interprétables et utilisables certaines directions de l'espace latent. Ainsi, il est possible d'avoir un certain contrôle sur la génération de variations d'une séquence donnée.

Distances de rang invariantes par transposition pour des séquences musicales

Posséder une distance entre séquences musicales est utile dans un grand nombre de tâches parmi lesquelles nous pouvons citer :

- la détection de plagiat
- l'analyse automatique de musique
- la génération automatique de musique.

Étant donné l'importance et l'omniprésence en musique de la notion de pattern ou de motif, le pré-requis pour une telle distance est d'être invariante par transposition. Aussi, il ne peut pas exister de distance entre séquences unique, utilisable sans distinction sur tous les corpus. En effet, la notion de similarité évolue en fonction des styles et des époques et deux séquences perceptuellement "proches" dans un style pourraient être fort éloignées dans un autre.

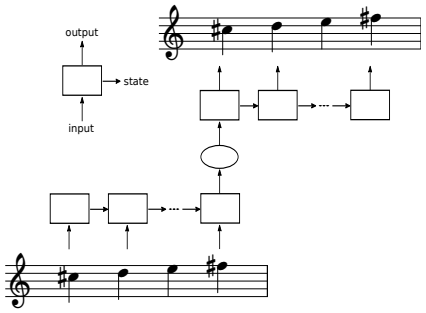


Figure 63: Auto-encodeur sur des séquences musicales

Cependant, la plupart des distances existantes pour la comparaison entre séquences musicales ne sont pas dépendantes d'un corpus. Je propose dans ce chapitre comment construire, dans un premier temps une distance adaptée à un corpus donné et, dans un second temps, de rendre cette distance invariante par transposition afin de pouvoir capturer la notion de motif indépendamment de la hauteur de la note initiale.

Cette distance est obtenue de la manière suivante : un auto-encodeur est tout d'abord entraîné sur les séquences du corpus, ce qui permet d'encoder chaque séquence musicale comme un vecteur latent de taille fixe. Étant données deux séquences, une distance de rang est alors utilisé entre les deux représentations latentes de ces séquences. Cette architecture est représentée dans la Fig. 63.

Cette distance jouit de bonnes propriétés mais n'est néanmoins pas invariante par transposition. Deux séquences, dont l'une est la transposition de l'autre posséderont deux représentations latentes différentes. L'objectif afin d'obtenir une distance invariante par transposition est alors de forcer ces deux représentations à être identiques. Ceci est fait grâce à un réseau de neurones dont l'architecture est montrée Fig. 64. L'objectif est, étant donné deux séquences, dont l'une est une transposition de l'autre, de prédire une transposition inconnue de cette séquence en connaissant uniquement sa note de départ. Afin d'obtenir une représentation invariante par transposition, la prédiction s'effectue à l'aide de la moyenne des représentations latentes des deux séquences d'entrée. Un terme de régularisation est également ajouté afin de réduire la distance entre les représentations latentes des séquences données en entrée du réseau.

L'application d'une distance de rang sur ces représentations latentes est particulièrement adaptée. En effet, même si les représentations latentes de deux transpositions d'une même séquence ne sont pas strictement égales, le fait de considérer une distance de rang sur ces vecteurs tend à gommer ces différences.

L'observation des plus proches voisins selon cette distance révèle sa pertinence d'un point de vue musical et son invariance par transposition. À terme, l'objectif est de pouvoir intégrer cette distance dans

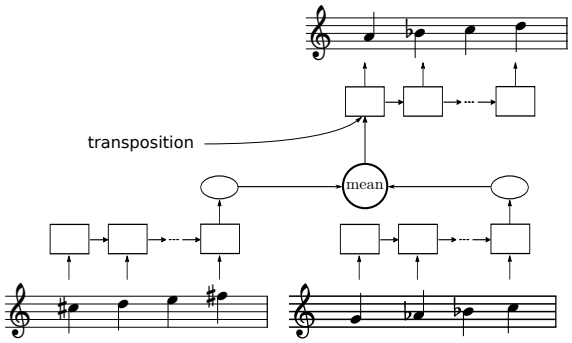


Figure 64: Architecture proposée permettant d'obtenir des représentations latentes invariantes par transposition. Les trois séquences musicales, en entrée comme en sortie, sont toutes des transpositions exactes d'une même séquence.

les algorithmes de composition existants afin de pouvoir imposer des contraintes de similarité entre différentes parties.

Anticipation-RNN: Génération interactive de musique sujette à des contraintes unaires

Ce chapitre introduit une architecture permettant de générer itérativement et de gauche à droite des séquences satisfaisant un ensemble de contraintes unaires. L'intérêt est l'efficacité de l'échantillonnage dans ce cas en comparaison des méthodes de Monte-Carlo par chaînes de Markov. Contrairement à l'échantillonnage classique de gauche à droite des réseaux récurrents qui ne permettent que de conditionner la génération par rapport au début de la séquence, Anticipation-RNN permet de conditionner la génération sur un ensemble arbitraire de contraintes unaires.

Étant donné un modèle génératif récurrent nommé *Token-RNN*, il s'agit donc de pouvoir échantillonner un sous-ensemble de séquences définies par des contraintes unaires avec les bonnes probabilités relatives. Les méthodes d'échantillonage avec rejet ont dans ce cas peu de chance d'aboutir. L'idée derrière l'architecture de Anticipation-RNN est d'introduire un second réseau récurrent nommé *Constraint-RNN*, traitant séquentiellement, mais de droite à gauche, l'ensemble des contraintes unaires fournies. Son but est alors de fournir au Token-RNN, celui-ci allant de gauche à droite, un résumé des contraintes unaires présentes dans le futur. L'architecture du Anticipation-RNN est présentée Fig. 65.

Les propriétés de ce réseau de neurones sont ensuite étudiées. Il est vérifié expérimentalement que cette architecture est capable de générer des séquences satisfaisant des contraintes unaires et que ces séquences auraient les mêmes probabilités relatives si elles avaient été échantillonnées avec un modèle sans contrainte. L'influence pré-

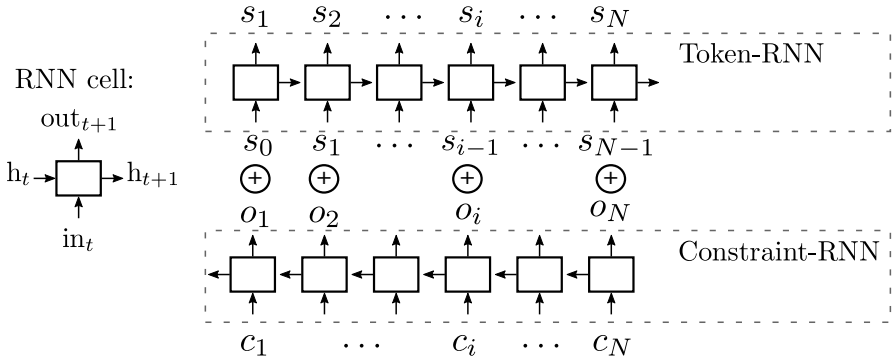


Figure 65: Architecture du Anticipation-RNN. l'objectif est de prédire la séquence décalée $(s_1, \ldots, s_N)$ sachant les contraintes $(c_1, \ldots, c_N)$ et la séquence initiale $(s_0, \ldots, s_{N-1})$. Le symbole "+" entouré désigne la concaténation en s_i et o_{i+1}

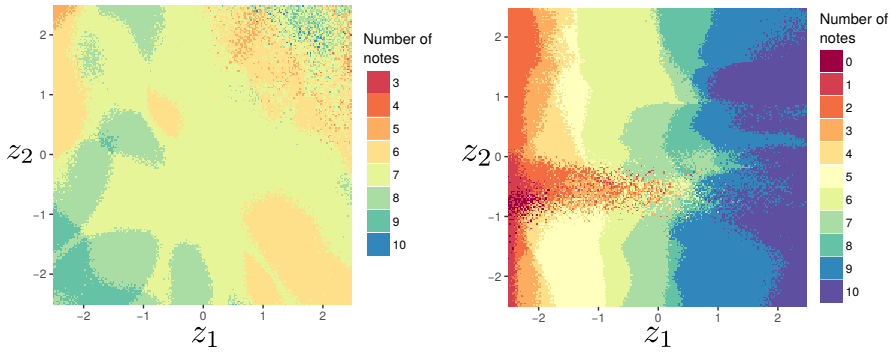
cise des contraintes unaires sur la génération est également mise en avant sur des exemples détaillés. Cela montre que ce modèle est effectivement capable d'anticiper les événements à venir tout en échantillonnant avec les bonnes probabilités.

La simplicité de cette approche est selon moi un atout majeur, puisque cela permet de rendre interactifs de nombreux modèles génératifs existants pour la musique symbolique.

GLSR-VAE: Régularisation géodésique de l'espace latent pour les auto-encodeurs variationnels

Ce dernier chapitre introduit une régularisation nommée *Régularisation géodésique de l'espace latent* pour les objectifs classiques utilisés pour l'entraînement des auto-encodeurs variationnels. Cette régularisation est générale et s'applique à de nombreuses architectures basées sur les auto-encodeurs variationnels.

L'intérêt de cette régularisation est de structurer l'espace latent des auto-encodeurs variationnels afin de rendre interprétables certaines directions de l'espace latent. Cela permet d'ajouter plus de contrôle aux générations produites par un auto-encodeur variationnel. En effet, la majorité des applications créatives de ces modèles commencent par entraîner un auto-encodeur variationnel pour ensuite effectuer des interpolations entre deux éléments. Il existe également certaines approches visant à trouver des directions interprétables a posteriori, notamment par le calcul de *vecteurs d'attribut*. Ces approches ne sont pas satisfaisantes à plusieurs titres : dans le premier cas, il existe une infinité de chemins effectuant une interpolation entre deux éléments et il n'est pas clair comment choisir la "meilleure" interpolation ; dans le second cas, on suppose implicitement que l'espace latent va structurer les données suivant des critères non définis durant l'apprentissage. Des méthodes existent pour choisir une inter-



(a) Sans régularisation GLSR (b) Avec régularisation GLSR

Figure 66: Visualisation du comportement de la quantité G (nombre de notes) sur un plan bidimensionnel de l'espace latent. L'axe z_1 correspond à la dimension où la régularisation est effective.

polation perceptuellement intéressante, mais elles sont coûteuses en ressources de calcul car elles reposent intrinsèquement sur des méthodes d'optimisation.

L'idée directrice dans ce chapitre est de se fixer certaines quantités d'intérêt sur nos données (cela peut-être par exemple le nombre de notes d'une séquence ou sa hauteur globale) et de relier les variations de ces quantités à des variations dans l'espace latent.

Supposons que l'on ait un auto-encodeur variationnel donné par $p(x,z) = p(z)p(x|z)$ ayant pour prior $p(z)$ et pour approximation de la distribution postérieure $q(z|x)$, ainsi que des mesures g_k sur nos données. On considère alors l'application suivante

$$G_k : z \mapsto \mathbf{E}_{p(x|z)}[g_k(x)] \tag{72}$$

qui relie l'espace latent à la valeur moyenne de nos mesures.

Il est possible de relier les variations dans l'espace latent aux variations des quantités $G_k(z)$ en imposant une contrainte sur les dérivées partielles des G_k. Idéalement, nous aimerions pouvoir fixer cette valeur à une constante. Dans les faits, on impose une loi normale sur la distribution des valeurs des dérivées partielles.

Les résultats expérimentaux montrent que cette démarche aboutit. La quantité d'intérêt considérée dans ce chapitre est le nombre de notes contenues dans la séquence. Un exemple de l'influence de cette régularisation est visible dans la Fig. 66. Sur cet exemple, pour un z_2 fixé, le nombre de notes croît lorsque la coordonnée z_1 croît.

Connaître la "direction" pour augmenter le nombre de notes d'une séquence est un moyen de générer, par exemples, des variations autour de cette séquence. Il est intéressant de souligner qu'ajouter cette régularisation semble décorréler les axes de l'espace latent.

Cette méthode permet donc de fournir la possibilité pour un utilisateur de générer des variations étant donné une séquence. Par rapport aux approches existantes qui consistent à effectuer des interpolations

dans l'espace latent et qui requièrent donc à un utilisateur de fournir deux séquences en entrée, il me semble que cette possibilité est plus intéressante et plus naturelle quant à la démarche compositionnelle.

BIBLIOGRAPHY

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[2] Claude Abromont and Eugène de Montalembert. *Guide des formes de la musique occidentale*. Fayard, Henri Lemoine, 2010.

[3] Julien Allali, Pascal Ferraro, Pierre Hanna, and Matthias Robine. "Polyphonic Alignment Algorithms for Symbolic Music Retrieval." In: *Auditory Display: 6th International Symposium, CMMR/ICAD 2009, Copenhagen, Denmark, May 18-22, 2009. Revised Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 466–482. ISBN: 978-3-642-12439-6. DOI: 10.1007/978-3-642-12439-6_24. URL: https://doi.org/10.1007/978-3-642-12439-6_24.

[4] Moray Allan and Christopher KI Williams. "Harmonising chorales by probabilistic inference." In: *Advances in neural information processing systems* 17 (2005), pp. 25–32.

[5] Shun-ichi Amari. *Information geometry and its applications*. Springer, 2016.

[6] Shun-ichi Amari and Hiroshi Nagaoka. *Methods of information geometry*. Vol. 191. American Mathematical Soc., 2007.

[7] Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, and Lucia Vadicamo. "Deep Permutations: Deep Convolutional Neural Networks and Permutation-Based Indexing." In: *International Conference on Similarity Search and Applications*. Springer. 2016, pp. 93–106.

[8] Barry C Arnold and David Strauss. "Pseudolikelihood estimation: some examples." In: *Sankhyā: The Indian Journal of Statistics, Series B* (1991), pp. 233–243.

[9] Francis R. Bach. "Structured sparsity-inducing norms through submodular functions." In: *Advances in Neural Information Processing Systems 23*. Ed. by J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta. Curran Associates, Inc., 2010, pp. 118–126. URL: http://papers.nips.cc/paper/3933-structured-sparsity-inducing-norms-through-submodular-functions.pdf.

[10] J.S. Bach. *389 Chorales (Choral-Gesange): SATB (German Language Edition)*. Kalmus Classic Edition. Alfred Publishing Company, 1985. ISBN: 9780769244204. URL: https://books.google.fr/books?id=U1-cAAAACAAJ.

[11] Yves Balmer, Thomas Lacôte, and Christopher Brent Murray. "Messiaen the Borrower: Recomposing Debussy through the Deforming Prism." In: *Journal of the American Musicological Society* 69.3 (2016), pp. 699–791. ISSN: 0003-0139. DOI: 10.1525/jams.2016.69.3.699. eprint: http://jams.ucpress.edu/content/69/3/699.full.pdf. URL: http://jams.ucpress.edu/content/69/3/699.

[12] Michele Basseville. "Distance measures for signal processing and pattern recognition." In: *Signal processing* 18.4 (1989), pp. 349–369.

[13] Y. Bengio, L. Yao, G. Alain, and P. Vincent. "Generalized Denoising Auto-Encoders as Generative Models." In: *ArXiv e-prints* (May 2013). arXiv: 1305.6663 [cs.LG].

[14] Yoshua Bengio et al. "Learning deep architectures for AI." In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127.

[15] M. Berglund, T. Raiko, M. Honkala, L. Kärkkäinen, A. Vetek, and J. Karhunen. "Bidirectional Recurrent Neural Networks as Generative Models - Reconstructing Gaps in Time Series." In: *ArXiv e-prints* (Apr. 2015). arXiv: 1504.01575 [cs.LG].

[16] A. Bietti and J. Mairal. "Group Invariance and Stability to Deformations of Deep Convolutional Representations." In: *ArXiv e-prints* (June 2017). arXiv: 1706.03078 [stat.ML].

[17] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription." In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. Edinburgh, Scotland, U.K., 2012, pp. 1159–1166.

[18] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. "Generating Sentences from a Continuous Space." In: *CoRR* abs/1511.06349 (2015). URL: http://arxiv.org/abs/1511.06349.

[19] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. "Deep Learning Techniques for Music Generation-A Survey." In: *arXiv preprint arXiv:1709.01620* (2017).

[20] Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. "Massive exploration of neural machine translation architectures." In: *arXiv preprint arXiv:1703.03906* (2017).

[21] Emilios Cambouropoulos, Tim Crawford, and Costas S Iliopoulos. "Pattern processing in melodic sequences: Challenges, caveats and prospects." In: *Computers and the Humanities* 35.1 (2001), pp. 9–21.

[22] Emilios Cambouropoulos, Maximos Kaliakatsos-Papakostas, and Costas Tsougras. *An idiom-independent representation of chords for computational music analysis and generation*. Ann Arbor, MI: Michigan Publishing, University of Michigan Library, 2014.

[23] Robert JS Cason and Daniel Müllensiefen. "Singing from the same sheet: computational melodic similarity measurement and copyright law." In: *International Review of Law, Computers & Technology* 26.1 (2012), pp. 25–36.

[24] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. *One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling*. Tech. rep. Google, 2013. URL: http://arxiv.org/abs/1312.3005.

[25] Shyh-Huei Chen and Edward H. Ip. "Behaviour of the Gibbs sampler when conditional distributions are potentially incompatible." In: *Journal of Statistical Computation and Simulation* 85.16 (2015), pp. 3266–3275. DOI: 10.1080/00949655.2014.968159. eprint: http://dx.doi.org/10.1080/00949655.2014.968159. URL: http://dx.doi.org/10.1080/00949655.2014.968159.

[26] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. "Variational lossy autoencoder." In: *arXiv preprint arXiv:1611.02731* (2016).

[27] Siddhartha Chib and Edward Greenberg. "Understanding the Metropolis-Hastings Algorithm." In: *The American Statistician* 49.4 (1995), pp. 327–335.

[28] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." In: *CoRR* abs/1406.1078 (2014). URL: http://arxiv.org/abs/1406.1078.

[29] François Chollet. *Keras*. https://github.com/fchollet/keras. 2015.

[30] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." In: *arXiv preprint arXiv:1412.3555* (2014).

[31] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. "A Recurrent Latent Variable Model for Sequential Data." In: *CoRR* abs/1506.02216 (2015). URL: http://arxiv.org/abs/1506.02216.

[32] Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. "Reducing overfitting in deep networks by decorrelating representations." In: *arXiv preprint arXiv:1511.06068* (2015).

[33] Taco S Cohen and Max Welling. "Steerable CNNs." In: *arXiv preprint arXiv:1612.08498* (2016).

[34] Taco Cohen and Max Welling. "Group equivariant convolutional networks." In: *International Conference on Machine Learning*. 2016, pp. 2990–2999.

[35] Florian Colombo, Alexander Seeholzer, and Wulfram Gerstner. "Deep Artificial Composer: A Creative Neural Network Model for Automated Melody Generation." In: *Computational Intelligence in Music, Sound, Art and Design: 6th International Conference, EvoMUSART 2017, Amsterdam, The Netherlands, April 19–21, 2017, Proceedings*. Cham: Springer International Publishing, 2017, pp. 81–96. ISBN: 978-3-319-55750-2. DOI: 10.1007/978-3-319-55750-2_6. URL: https://doi.org/10.1007/978-3-319-55750-2_6.

[36] Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. "Recurrent batch normalization." In: *arXiv preprint arXiv:1603.09025* (2016).

[37] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[38] Michael Scott Cuthbert and Christopher Ariza. "music21: A toolkit for computer-aided musicology and symbolic music data." In: *International Society for Music Information Retrieval* (2010).

[39] Bin Dai, Yu Wang, John Aston, Gang Hua, and David P. Wipf. "Hidden Talents of the Variational Autoencoder." In: *CoRR* abs/1706.05148 (2017). arXiv: 1706.05148. URL: http://arxiv.org/abs/1706.05148.

[40] Laurent David, Mathieu Giraud, Richard Groult, Florence Levé, and Corentin Louboutin. "Vers une analyse automatique des formes sonates." In: *Journées d'Informatique Musicale (JIM 2014)*. 2014, pp. 113–118.

[41] Christopher De Sa, Kunle Olukotun, and Christopher Ré. "Ensuring Rapid Mixing and Low Bias for Asynchronous Gibbs Sampling." In: *arXiv preprint arXiv:1602.07415* (2016).

[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database." In: *CVPR09*. 2009.

[43] Michel Marie Deza and Elena Deza. "Encyclopedia of distances." In: *Encyclopedia of Distances*. Springer, 2009, pp. 1–583.

[44] Kemal Ebcioglu. "An Expert System for Harmonizing Four-Part Chorales." In: *Computer Music Journal* 12.3 (1988), pp. 43–51. ISSN: 01489267, 15315169. URL: http://www.jstor.org/stable/3680335.

[45] Magnus Ekeberg, Cecilia Lövkvist, Yueheng Lan, Martin Weigt, and Erik Aurell. "Improved contact prediction in proteins: using pseudolikelihoods to infer Potts models." In: *Physical Review E* 87.1 (2013), p. 012707.

[46] Manfred Eppe, Roberto Confalonieri, Ewen Maclean, Maximos Kaliakatsos, Emilios Cambouropoulos, Marco Schorlemmer, Mihai Codescu, and Kai-Uwe Kühnberger. "Computational invention of cadences and chord progressions by conceptual chord-blending." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2015.

[47] O. Fabius and J. R. van Amersfoort. "Variational Recurrent Auto-Encoders." In: *ArXiv e-prints* (Dec. 2014). arXiv: 1412.6581 [stat.ML].

[48] Jose D Fernández and Francisco Vico. "AI methods in algorithmic composition: A comprehensive survey." In: *Journal of Artificial Intelligence Research* (2013), pp. 513–582.

[49] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. "Sequential neural models with stochastic layers." In: *Advances in Neural Information Processing Systems*. 2016, pp. 2199–2207.

[50] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin, 2001.

[51] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "A note on the group lasso and a sparse group lasso." In: *arXiv preprint arXiv:1001.0736* (2010).

[52] Stuart Geman and Donald Geman. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1984), pp. 721–741.

[53] Robert Gens and Pedro M Domingos. "Deep Symmetry Networks." In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 2537–2545. URL: http://papers.nips.cc/paper/5424-deep-symmetry-networks.pdf.

[54] Felix A Gers and Jürgen Schmidhuber. "Recurrent nets that time and count." In: *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*. Vol. 3. IEEE. 2000, pp. 189–194.

[55] Andrew Gibiansky, Sercan Arik, Gregory Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. "Deep Voice 2: Multi-Speaker Neural Text-to-Speech." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 2966–2974. URL: http://papers.nips.cc/paper/6889-deep-voice-2-multi-speaker-neural-text-to-speech.pdf.

[56] Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. "Computational fugue analysis." In: *Computer Music Journal* 39.2 (2015), pp. 77–96.

[57] Kratarth Goel, Raunaq Vohra, and J. K. Sahoo. "Polyphonic music generation by modeling temporal dependencies using a RNN-DBN." In: *Proceedings of the International Conference on Artificial Neural Networks*. Springer, 2014, pp. 217–224.

[58] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[59] Maarten Grachten, Josep-Lluıs Arcos, and Ramon López De Mántaras. "Melodic similarity: Looking for a good abstraction level." In: *representations* 2 (2004), p. 7.

[60] Alex Graves. "Supervised sequence labelling." In: *Supervised sequence labelling with recurrent neural networks*. Springer, 2012, pp. 5–13.

[61] K. Gregor, I. Danihelka, A. Graves, D. Jimenez Rezende, and D. Wierstra. "DRAW: A Recurrent Neural Network For Image Generation." In: *ArXiv e-prints* (Feb. 2015). arXiv: 1502.04623 [cs.CV].

[62] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. "DeepBach: a Steerable Model for Bach Chorales Generation." In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2017, pp. 1362–1371. URL: http://proceedings.mlr.press/v70/hadjeres17a.html.

[63] Gaëtan Hadjeres, Jason Sakellariou, and François Pachet. "Style Imitation and Chord Invention in Polyphonic Music with Exponential Families." In: *CoRR* abs/1609.05152 (2016). URL: http://arxiv.org/abs/1609.05152.

[64] Pierre Hanna, Pascal Ferraro, and Matthias Robine. "On optimizing the editing algorithms for evaluating similarity between monophonic musical sequences." In: *Journal of New Music Research* 36.4 (2007), pp. 267–279.

[65] David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. "Dependency networks for inference, collaborative filtering, and data visualization." In: *Journal of Machine Learning Research* 1.Oct (2000), pp. 49–75.

[66] Hermann Hild, Johannes Feulner, and Wolfram Menzel. "HARMONET: A neural net for harmonizing chorales in the style of JS Bach." In: *Advances in neural information processing systems*. 1992, pp. 267–274.

[67] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." In: *Neural Computation* 18.7 (July 2006), pp. 1527–1554.

[68] Geoffrey Hinton. "A practical guide to training restricted Boltzmann machines." In: *Momentum* 9.1 (2010), p. 926.

[69] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[70] Z. Hu, Z. Yang, R. Salakhutdinov, and E. P. Xing. "On Unifying Deep Generative Models." In: *ArXiv e-prints* (June 2017). arXiv: 1706.00550 [cs.LG].

[71] Allen Huang and Raymond Wu. *Deep Learning for Music*. arXiv:1606.04930v1. June 2016.

[72] Aapo Hyvärinen. "Estimation of non-normalized statistical models by score matching." In: *Journal of Machine Learning Research*. 2005, pp. 695–709.

[73] Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck. *Tuning Recurrent Neural Networks with Reinforcement Learning*. arXiv:1611.02796. Nov. 2016.

[74] Daniel Johnson. *Composing Music With Recurrent Neural Networks*. http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/. Aug. 2015.

[75] Junbo, Zhao, Y. Kim, K. Zhang, A. M. Rush, and Y. LeCun. "Adversarially Regularized Autoencoders for Generating Discrete Structures." In: *ArXiv e-prints* (June 2017). arXiv: 1706.04223 [cs.LG].

[76] Maximos Kaliakatsos-Papakostas and Emilios Cambouropoulos. "Probabilistic harmonization with fixed intermediate chord constraints." In: *ICMC*. 2014.

[77] Frank P Kelly. *Reversibility and stochastic networks*. Cambridge University Press, 2011.

[78] D. P Kingma and M. Welling. "Auto-Encoding Variational Bayes." In: *ArXiv e-prints* (Dec. 2013). arXiv: 1312.6114 [stat.ML].

[79] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014).

[80] Charles Koechlin. *Traité de l'harmonie: en 3 volumes*. Vol. 1. Eschig, 1928.

[81] W. Krauth. *Statistical Mechanics: Algorithms and Computations*. Oxford Master Series in Physics. Oxford University Press, UK, 2006. ISBN: 9780191523328. URL: https://books.google.fr/books?id=EnabPPmmS4sC.

[82] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[83] David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, Aaron Courville, et al. "Zoneout: Regularizing rnns by randomly preserving hidden activations." In: *arXiv preprint arXiv:1606.01305* (2016).

[84] Alex Lamb, Vincent Dumoulin, and Aaron Courville. "Discriminative regularization for generative models." In: *arXiv preprint arXiv:1602.03220* (2016).

[85] Hugo Larochelle and Iain Murray. "The Neural Autoregressive Distribution Estimator." In:

[86] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. "Autoencoding beyond pixels using a learned similarity metric." In: *arXiv preprint arXiv:1512.09300* (2015).

[87] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. *Imposing Higher-level Structure in Polyphonic Music Generation using Convolutional Restricted Boltzmann Machines and Constraints*. arXiv:1612.04742v2. Dec. 2016.

[88] Jay Yoon Lee, Michael Wick, Jean-Baptiste Tristan, and Jaime G. Carbonell. "Enforcing Constraints on Outputs with Unconstrained Inference." In: *CoRR* abs/1707.08608 (2017). arXiv: 1707.08608. URL: http://arxiv.org/abs/1707.08608.

[89] Kjell Lemström and Anna Pienimäki. "On comparing edit distance and geometric frameworks in content-based retrieval of symbolically encoded polyphonic music." In: *Musicae Scientiae* 11.1_suppl (2007), pp. 135–152. DOI: 10.1177/102986490701100106. eprint: http://dx.doi.org/10.1177/102986490701100106. URL: http://dx.doi.org/10.1177/102986490701100106.

[90] Kjell Lemström and Esko Ukkonen. "Including interval encoding into edit distance based music comparison and retrieval." In: *In Proceedings of the AISB'2000 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*. 2000.

[91] Vladimir I Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals." In: *Soviet physics doklady*. Vol. 10. 1966, pp. 707–710.

[92] Xiangang Li and Xihong Wu. "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition." In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 4520–4524.

[93] Feynman Liang. *BachBot: Automatic composition in the style of Bach chorales*. 2016.

[94] William Paul Livant. "L. A. Hiller and L. M. Isaacson. Experimental music. New York: McGraw-hill, 1960." In: *Behavioral Science* 6.2 (1961), pp. 159–160. ISSN: 1099-1743. DOI: 10.1002/bs.3830060211. URL: http://dx.doi.org/10.1002/bs.3830060211.

[95] Qi Lyu, Zhiyong Wu, Jun Zhu, and Helen Meng. "Modelling high-dimensional sequences with LSTM-RTRBM: application to polyphonic music generation." In: *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press. 2015, pp. 4138–4139.

[96] MIDI Manufacturers Association (MMA). *MIDI Specifications*. https://www.midi.org/specifications. Accessed on 14/04/2017.

[97] Lars Maaløe, Marco Fraccaro, and Ole Winther. "Semi-Supervised Generation with Cluster-aware Generative Models." In: *arXiv preprint arXiv:1704.00637* (2017).

[98] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. "Adversarial autoencoders." In: *arXiv preprint arXiv:1511.05644* (2015).

[99] Siddharth Manay, Daniel Cremers, Byung-Woo Hong, Anthony J Yezzi, and Stefano Soatto. "Integral invariants for shape matching." In: *IEEE Transactions on pattern analysis and machine intelligence* 28.10 (2006), pp. 1602–1618.

[100] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. "The Stanford CoreNLP Natural Language Processing Toolkit." In: *ACL 2014* (2014), p. 55.

[101] Elman Mansimov, Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. "Generating Images from Captions with Attention." In: *CoRR* abs/1511.02793 (2015). arXiv: 1511.02793. URL: http://arxiv.org/abs/1511.02793.

[102] David Meredith, Kjell Lemström, and Geraint A. Wiggins. "Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music." In: *Journal of New Music Research* 31.4 (2002), pp. 321–345. DOI: 10.1076/jnmr.31.4.321.14162. eprint: http://www.tandfonline.com/doi/pdf/10.1076/jnmr.31.4.321.14162. URL: http://www.tandfonline.com/doi/abs/10.1076/jnmr.31.4.321.14162.

[103] Marc Mezard and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

[104] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space." In: *ArXiv e-prints* (Jan. 2013). arXiv: 1301.3781 [cs.CL].

[105] Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. "Learning longer memory in recurrent neural networks." In: *arXiv preprint arXiv:1412.7753* (2014).

[106] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning." In: *arXiv preprint arXiv:1312.5602* (2013).

[107] Marcel Mongeau and David Sankoff. "Comparison of musical sequences." In: *Computers and the Humanities* 24.3 (1990), pp. 161–175.

[108] M. C. Mozer. "Neural network composition by prediction: Exploring the benefits of psychophysical constraints and multiscale processing." In: *Cognitive Science* 6 (1994), pp. 247–280.

[109] Michael C Mozer. "A focused back-propagation algorithm for temporal pattern recognition." In: *Complex systems* 3.4 (1989), pp. 349–381.

[110] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines." In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, pp. 807–814.

[111] Eugene Narmour. *The analysis and cognition of melodic complexity: The implication-realization model*. University of Chicago Press, 1992.

[112] Frank Nielsen and Sylvain Boltz. "The Burbea-Rao and Bhattacharyya centroids." In: *IEEE Transactions on Information Theory* 57.8 (2011), pp. 5455–5466.

[113] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." In: *arXiv preprint arXiv:1601.06759* (2016).

[114] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. "Wavenet: A generative model for raw audio." In: *arXiv preprint arXiv:1609.03499* (2016).

[115] Ferdinand Österreicher and Igor Vajda. "A new class of metric divergences on probability spaces and its applicability in statistics." In: *Annals of the Institute of Statistical Mathematics* 55.3 (2003), pp. 639–653. ISSN: 1572-9052. DOI: 10.1007/BF02517812. URL: https://doi.org/10.1007/BF02517812.

[116] F. Pachet. "A Joyful Ode to Automatic Orchestration." In: *ACM Transactions on Intelligent Systems and Technology, Special Issue on Intelligent Music Systems and Applications* (2016). (invited contribution; to appear).

[117] F. Pachet and P. Roy. "Non-Conformant Harmonization: the Real Book in the Style of Take 6." In: *5th International Conference on Computational Creativity (ICCC 2014)*. Ljubljiana (Slovenia), 2014, pp. 100–107.

[118] François Pachet, Jeff Suzda, and Daniel Martín. "A Comprehensive Online Database of Machine-Readable Leadsheets for Jazz Standards." In: *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR'2013)*. Ed. by Alceu de Souza Britto Junior, Fabien Gouyon, and Simon Dixon. Curitiba, Brazil, Nov. 2013, pp. 275–280.

[119] Alexandre Papadopoulos and Pierre Roy. "Exact Sampling for Regular and Markov Constraints With Belief Propagation." In: *Constraint Programming, CP* (2015), pp. 1–11. URL: http://csl.sony.fr/downloads/papers/2015/papadopoulos-15b.pdf.

[120] Alexandre Papadopoulos, Pierre Roy, and François Pachet. "Assisted Lead Sheet Composition Using FlowComposer." In: *Principles and Practice of Constraint Programming: 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*. Ed. by Michel Rueher. Cham: Springer International Publishing, 2016, pp. 769–785. ISBN: 978-3-319-44953-1. DOI: 10.1007/978-3-319-44953-1_48. URL: http://dx.doi.org/10.1007/978-3-319-44953-1_48.

[121] Alexandre Papadopoulos, Pierre Roy, and François Pachet. "Assisted Lead Sheet Composition Using FlowComposer." In: *CP*. Vol. 9892. Lecture Notes in Computer Science. Springer, 2016, pp. 769–785.

[122] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. "How to Construct Deep Recurrent Neural Networks." In: *ArXiv e-prints* (Dec. 2013). arXiv: 1312.6026.

[123] Jeremy Pickens and Costas S Iliopoulos. "Markov Random Fields and Maximum Entropy Modeling for Music Information Retrieval." In: *ISMIR*. Citeseer. 2005, pp. 207–214.

[124] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. "Variational Autoencoder for Deep Learning of Images, Labels and Captions." In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 2352–2360. URL: http://papers.nips.cc/paper/6528-variational-autoencoder-for-deep-learning-of-images-labels-and-captions.pdf.

[125] *PyTorch*. https://github.com/pytorch/pytorch. 2016.

[126] Jean-Philippe Rameau. *Traité de l'harmonie réduite à ses principes naturels.* Imp. de J.-B.-C. Ballard, 1722.

[127] Pradeep Ravikumar, Martin J Wainwright, John D Lafferty, et al. "High-dimensional Ising model selection using l1-regularized logistic regression." In: *The Annals of Statistics* 38.3 (2010), pp. 1287–1319.

[128] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. "On stochastic optimal control and reinforcement learning by approximate inference." In: *Robotics: science and systems*. 2012.

[129] Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. "Generative Adversarial Text to Image Synthesis." In: *CoRR* abs/1605.05396 (2016). arXiv: 1605.05396. URL: http://arxiv.org/abs/1605.05396.

[130] Scott E Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. "Learning what and where to draw." In: *Advances in Neural Information Processing Systems*. 2016, pp. 217–225.

[131] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck. "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music." In: *ArXiv e-prints* (Mar. 2018). arXiv: 1803.05428 [cs.LG].

[132] Pierre Roy, Alexandre Papadopoulos, and François Pachet. "Sampling Variations of Lead Sheets." In: *CoRR* abs/1703.00760 (2017). URL: http://arxiv.org/abs/1703.00760.

[133] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge." In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[134] J. Sakellariou, F. Tria, V. Loreto, and F. Pachet. "Maximum Entropy Model for Melodic Patterns." In: *ICML Workshop on Constructive Machine Learning*. Paris (France), 2015.

[135] J. Sakellariou, F. Tria, V. Loreto, and F. Pachet. "Maximum entropy models capture melodic styles." In: *ArXiv e-prints* (Oct. 2016). arXiv: 1610.03414 [stat.ML].

[136] Jason Sakellariou, Francesca Tria, Vittorio Loreto, and Francois Pachet. "Maximum entropy models capture melodic styles." In: *Scientific Reports* 7.1 (2017), p. 9172. ISSN: 2045-2322. DOI: 10.1038/s41598-017-08028-4. URL: https://doi.org/10.1038/s41598-017-08028-4.

[137] Mike Schuster and Kuldip K Paliwal. "Bidirectional recurrent neural networks." In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[138] Dmitriy Serdyuk, Nan Rosemary Ke, Alessandro Sordoni, Chris Pal, and Yoshua Bengio. "Twin Networks: Using the Future as a Regularizer." In: *CoRR* abs/1708.06742 (2017). arXiv: 1708.06742. URL: http://arxiv.org/abs/1708.06742.

[139] S. Shabanian, D. Arpit, A. Trischler, and Y. Bengio. "Variational Bi-LSTMs." In: *ArXiv e-prints* (Nov. 2017). arXiv: 1711.05717 [stat.ML].

[140] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *arXiv preprint arXiv:1409.1556* (2014).

[141] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning Structured Output Representation using Deep Conditional Generative Models." In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 3483–3491. URL: http://papers.nips.cc/paper/5775-learning-structured-output-representation-using-deep-conditional-generative-models.pdf.

[142] Greg J Stephens and William Bialek. "Toward a statistical mechanics of four letter words." In: *arXiv preprint arXiv:0801.0253* (2007).

[143] Bob L. Sturm and João Felipe Santos. *The Endless Traditional Music Session*. http://www.eecs.qmul.ac.uk/%7Esturm/research/RNNIrishTrad/. Accessed on 21/12/2016.

[144] Bob L. Sturm, Oded Ben-Tal, Ùna Monaghan, Nick Collins, Dorien Herremans, Elaine Chew, Gaëtan Hadjeres, Emmanuel Deruty, and Françcois Pachet. *Machine Learning in Music Practice: A Case Study*. Submitted to Journal of New Music Research.

[145] Bob L. Sturm, João Felipe Santos, Oded Ben-Tal, and Iryna Korshunova. *Music transcription modelling and composition using deep learning*. arXiv:1604.08723v1. Apr. 2016.

[146] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks." In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[147] Lucas Theis, Aäron van den Oord, and Matthias Bethge. *A note on the evaluation of generative models*. arXiv:1511.01844. 2015.

[148] Tijmen Tieleman. "Training restricted Boltzmann machines using approximations to the likelihood gradient." In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 1064–1071.

[149] Alexey Tikhonov and Ivan P. Yamshchikov. "Music generation with variational recurrent autoencoder supported by history." In: *CoRR* abs/1705.05458 (2017). URL: http://arxiv.org/abs/1705.05458.

[150] Peter M. Todd. "A connectionist approach to algorithmic composition." In: *Computer Music Journal* 13.4 (1989), pp. 27–43.

[151] Godfried T Toussaint. "Algorithmic, geometric, and combinatorial problems in computational music theory." In: *Proceedings of X Encuentros de Geometria Computacional* (2003), pp. 101–107.

[152] Paul Upchurch, Jacob Gardner, Kavita Bala, Robert Pless, Noah Snavely, and Kilian Weinberger. "Deep feature interpolation for image content changes." In: *arXiv preprint arXiv:1611.05507* (2016).

[153] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." In: *Journal of Machine Learning Research* 11.Dec (2010), pp. 3371–3408.

[154] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. "Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge." In: *CoRR* abs/1609.06647 (2016). arXiv: 1609.06647. URL: http://arxiv.org/abs/1609.06647.

[155] M. Vucelja. "Lifting – A nonreversible Markov chain Monte Carlo Algorithm." In: *ArXiv e-prints* (Dec. 2014). arXiv: 1412.8762 [cond-mat.stat-mech].

[156] Christian Walder. "Modelling Symbolic Music: Beyond the Piano Roll." In: *Asian Conference on Machine Learning*. 2016, pp. 174–189.

[157] Chris Walshaw. *abc notation home page*. http://abcnotation.com. Accessed on 21/12/2016.

[158] Paul J Werbos. "Backpropagation through time: what it does and how to do it." In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[159] Raymond P. Whorley and Darrell Conklin. "Music Generation from Statistical Models of Harmony." In: *Journal of New Music Research* 45.2 (2016), pp. 160–183. DOI: 10.1080/09298215.2016.1173708. eprint: http://dx.doi.org/10.1080/09298215.2016.1173708. URL: http://dx.doi.org/10.1080/09298215.2016.1173708.

[160] Raymond P Whorley, Geraint A Wiggins, Christophe Rhodes, and Marcus T Pearce. "Multiple viewpoint systems: Time complexity and the construction of domains for complex musical viewpoints in the harmonization problem." In: *Journal of New Music Research* 42.3 (2013), pp. 237–266.

[161] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. "Harmonic Networks: Deep Translation and Rotation Equivariance." In: *CoRR* abs/1612.04642 (2016). URL: http://arxiv.org/abs/1612.04642.

[162] X. Yan, J. Yang, K. Sohn, and H. Lee. "Attribute2Image: Conditional Image Generation from Visual Attributes." In: *ArXiv e-prints* (Dec. 2015). arXiv: 1512.00570 [cs.LG].