DeepBach: a Steerable Model for Bach Chorales Generation

Gaëtan Hadjeres [1 2] François Pachet [1 2] Frank Nielsen [3]

Abstract

This paper introduces DeepBach, a graphical model aimed at modeling polyphonic music and specifically hymn-like pieces. We claim that, after being trained on the chorale harmonizations by Johann Sebastian Bach, our model is capable of generating highly convincing chorales in the style of Bach. DeepBach's strength comes from the use of pseudo-Gibbs sampling coupled with an adapted representation of musical data. This is in contrast with many automatic music composition approaches which tend to compose music sequentially. Our model is also steerable in the sense that a user can constrain the generation by imposing positional constraints such as notes, rhythms or cadences in the generated score. We also provide a plugin on top of the MuseScore music editor making the interaction with DeepBach easy to use.

1. 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. The corpus of the chorale harmonizations by Johann Sebastian Bach is remarkable by its homogeneity and its size (389 chorales in (Bach, 1985)). All these short pieces (approximately one minute long) are written for a four-part chorus (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. the three lower parts (alto, tenor and bass) accompanying the soprano (the highest part) are composed, see Fig.1 for an example.

巴哈逝世300 年 389聖詩

Moreover, since the aim of reharmonizing a melody is to give more power or new insights to its text, the lyrics have to be understood clearly. We say that voices are in *homophony*, i.e. they articulate syllables simultaneously. This implies characteristic rhythms, variety of harmonic ideas as well as characteristic melodic movements which make 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 proposed by (Ebcioglu, 1988) 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.

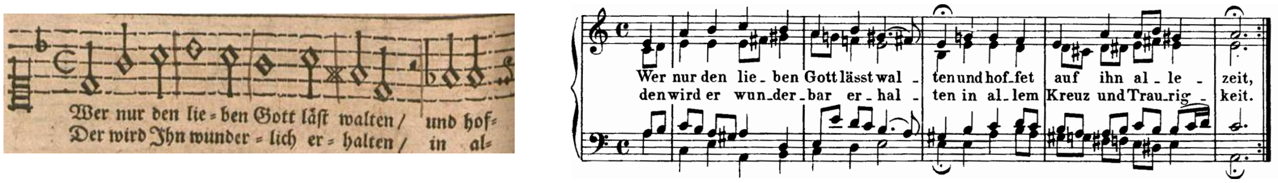
1988年本篇結果很差。製作出來的不過是偶有巴哈作品的某一段擷取

A neural-network-based solution was later developed by (Hild et al., 1992). 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 (Allan & Williams, 2005), but uses Hidden Markov Models (HMMs) instead of neural networks. Chords are represented as lists of intervals and form the states of the Markov mod-

[1] LIP6, Université Pierre et Marie Curie [2] Sony CSL, Paris [3] Sony CSL, Japan. Correspondence to: Gaëtan Hadjeres <gaetan.hadjeres@etu.upmc.fr>, François Pachet <pachetcsl@gmail.com>, Frank Nielsen <Frank.Nielsen@acm.org>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

[2] https://www.youtube.com/watch?v=73WF0M99vlg


Wer nur den lie-ben Gott läſſt walten/ und hof-
Der wird Ihn wunder-lich er-halten/ in al-
Wer nur den lie_ben Gott lässt wal_ ten und hof_fet auf ihn al_le_ zeit,
den wird er wun_der_bar er_hal_ ten in al_lem Kreuz und Trau_rig_ keit.

(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 1. Two versions of "Wer nur den lieben Gott läßt walten". The original melody (a) and its reharmonization (b) by Johann Sebastian Bach (BWV 434) [2].

els. These approaches produce interesting results even if they both use expert knowledge and bias the generation by imposing their compositional process. In (Whorley et al., 2013; Whorley & Conklin, 2016), authors elaborate on those 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 and the resort to hand-crafted criteria to assess the quality of the generated sequences might rule out many musically-interesting solutions.

Recently, agnostic approaches (requiring no knowledge about harmony, Bach's style or music) using neural networks have been investigated with promising results. In (Boulanger-Lewandowski et al., 2012), 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 ((Hochreiter & Schmidhuber, 1997; Mikolov et al., 2014)) or GRUs (Gated Recurrent Units) have been developed by (Lyu et al., 2015) and (Chung et al., 2014) 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 (Sakellariou et al., 2015; Sakellariou et al., 2016) for monophonic music and later generalized to polyphony in (Hadjeres et al., 2016). 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 (Liang, 2016), 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. 2) which are indicators of the structure of the chorales.

In this paper we introduce DeepBach, a dependency network (Heckerman et al., 2000) 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 (Papadopoulos et al., 2016) for leadsheets.

In Sect. 2 we present the DeepBach model for four-part chorale generation. We discuss in Sect. 3 the results of an experimental study we conducted to assess the quality of our model. Finally, we provide generated examples in Sect. 4.3 and elaborate on the possibilities offered by our interactive music composition editor in Sect. 4. All examples can be heard on the accompanying web page[3] and the code of our implementation is available on GitHub[4]. 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.

[3] https://sites.google.com/site/deepbachexamples/

[4] https://github.com/Ghadjeres/DeepBach

2. DeepBach

In this paper we introduce a generative model which takes into account the distinction between voices. Sect. 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. 2.2 presents the model's architecture and Sect. 2.3 our generation method. Finally, Sect. 2.4 provides implementation details and indicates how we preprocessed the corpus of Bach chorale harmonizations.

2.1. Data Representation

2.1.1. Notes and Voices

We use MIDI pitches to encode notes and choose to model voices separately. 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 $t \in [T]$[5], where T is the duration piece (in sixteenth notes).

We adopt the standard notation [N] to denote the set of inte- gers {1,...,N} for any integer N.

2.1.2. Rhythm

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. 2.3.4).

2.1.3. Metadata

The music sheet (Fig. 1b) 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. 2),
  延音
- current key,
- current key signature,
- current mode (major/minor/dorian).

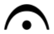


Figure 2. Fermata symbol.

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. 2, 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.

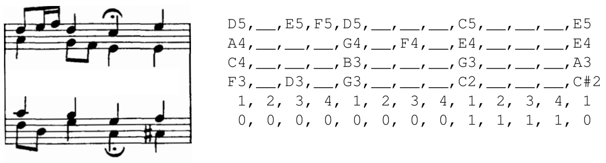
2.1.4. Chorale

We represent a chorale as a couple

$$(\mathcal{V}, \mathcal{M}) \tag{1}$$

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.



```
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 3. 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.

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

2.2. 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. 1 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 $\theta_{i,t}$)

$$\left\{p_{i,t}(V_i^t | V_{\setminus i,t}, \mathcal{M}, \theta_{i,t})\right\}_{i\in[4],t\in[T]}, \qquad (2)$$

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:

$$\max_{\theta_i} \sum_t \log p_i(\mathcal{V}_i^t | \mathcal{V}_{\setminus i,t}, \mathcal{M}, \theta_i), \qquad \text{for } i \in [4], \qquad (3)$$

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 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 (Pascanu et al., 2013), 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 4 shows a graphical representation for one of these models. Details are provided in Sect. 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."

2.3. Generation

2.3.1. 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 sampling procedure (Geman & Geman, 1984) on the difference that the conditional distributions are potentially incompatible (Chen & Ip, 2015). This means that the conditional distributions of Eq. (2) 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 (Kelly, 2011): it is thus not reversible and cannot converge to a joint distribution whose conditional distributions match the ones used for sampling.

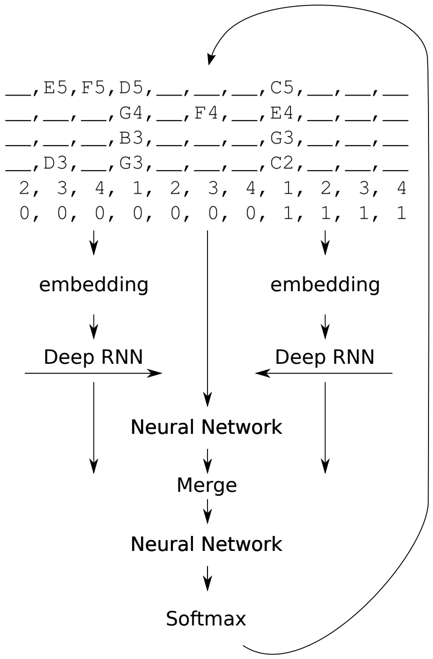


Figure 4. Graphical representations of DeepBach's neural network architecture for the soprano prediction p_1.

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 (Heckerman et al., 2000). Furthermore, nonreversible MCMC algorithms can in particular cases be better at sampling that reversible Markov Chains (Vucelja, 2014).

2.3.2. 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

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:** $\mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{V}_4)$
```

and 4 in step (3) 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.

2.3.3. 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 (De Sa et al., 2016) (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 (Krauth, 2006)) to appropriately choose all the time indexes at which we parallelly 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.

2.3.4. 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.

2.4. Implementation Details

We implemented DeepBach using Keras (Chollet, 2015) with the Tensorflow (Abadi et al., 2015) backend. We used the database of chorale harmonizations by J.S. Bach included in the music21 toolkit (Cuthbert & Ariza, 2010). 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. 4, we model only *local* interactions between a note $\mathcal{V}_i^t$ and its context $(\mathcal{V}_{\setminus i,t}, \mathcal{M})$ 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. 3 and examples in Sect. 4.3 were obtained with $\Delta t = 16$. We chose as the "neural network brick" in Fig. 4 a neural network with one hidden layer of size 200 and ReLU (Nair & Hinton, 2010) nonlinearity and as the "Deep RNN brick" two stacked LSTMs (Hochreiter & Schmidhuber, 1997; Mikolov et al., 2014), each one being of size 200 (see Fig. 2 (f) in (Li & Wu, 2015)). 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.

3. Experimental Results

We evaluated the quality of our model with an online test conducted on human listeners.

3.1. Setup

For the parameters used in our experiment, see Sect 2.4. We compared our model with two other models: a Maximum Entropy model (MaxEnt) as in (Hadjeres et al., 2016) and a Multilayer Perceptron (MLP) model.

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

$$p_i(\mathcal{V}_i^t | \mathcal{V}_{\setminus i,t}, \mathcal{M}, A_i, b_i) = \text{Softmax}(AX + b) \qquad (4)$$

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

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

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

The Multilayer Perceptron model 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 (Nair & Hinton, 2010) nonlinearity.

All models are local and have the same scope Δt, see Sect. 2.4.

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

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

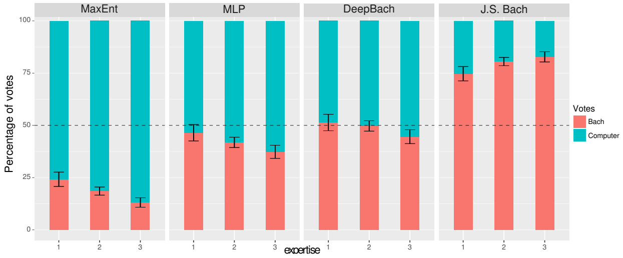


Figure 5. Results of the "Bach or Computer" experiment. The figure shows the distribution of the votes between "Computer" (blue bars) and "Bach" (red bars) for each model and each level of expertise of the voters (from 1 to 3), see Sect. 3.2 for details.

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[6] 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.

3.2. Discrimination Test: "Bach or Computer" experiment

Subjects were presented series of only one musical extract together with the binary choice "Bach" or "Computer". Fig. 5 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 accurate when the level of musical expertise is higher. When presented a DeepBach-generated

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

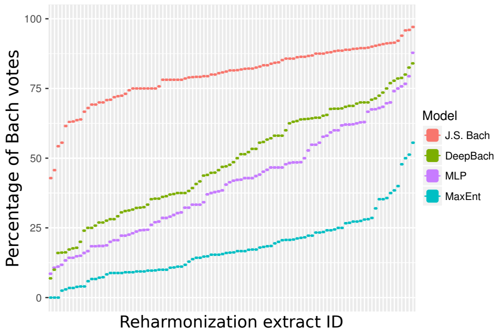


Figure 6. Results of the "Bach or Computer" experiment. The figure shows the percentage of votes for Bach for each of the 100 extracts for each model. For each model, a specific order for the x-axis is chosen so that the percentage of Bach votes is an increasing function of the x variable, see Sect. 3.2 for details.

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. 6 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.

4. Interactive composition

4.1. Description

We developed a plugin on top of the MuseScore music editor allowing a user to call DeepBach on any rectangular region. Even if the interface is minimal (see Fig.7), 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.

4.2. Adapting the model

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

4.2.1. Note encoding

We changed the MIDI encoding of the notes to a *full name* encoding of the notes. Indeed, some information is lost when reducing a music sheet to its MIDI representation since we cannot differentiate between two *enharmonic notes* (notes that sound the same but that are written differently e.g. F# and Gb). This difference in Bach chorales is unambiguous and it is thus natural to consider the *full name* of the notes, like C#3, Db3 or E#4. From a machine learning point of view, these notes would appear in totally different contexts. This improvement enables the model to generate notes with the correct spelling, which is important when we focus on the music sheet rather than on its audio rendering.

4.2.2. 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.

4.3. Generation examples

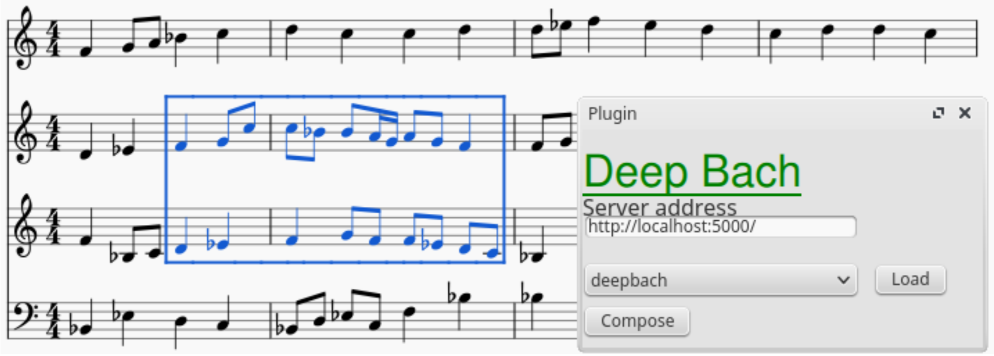
We now provide and comment on examples of chorales generated using the DeepBach plugin. Our aim is to show the quality of the solutions produced by DeepBach. For these examples, no note was set by hand and we asked DeepBach to generate regions longer than one bar and covering all four voices.

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. 4.2, DeepBach also learned the correct spelling of the notes. Among examples in Fig. 8, 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

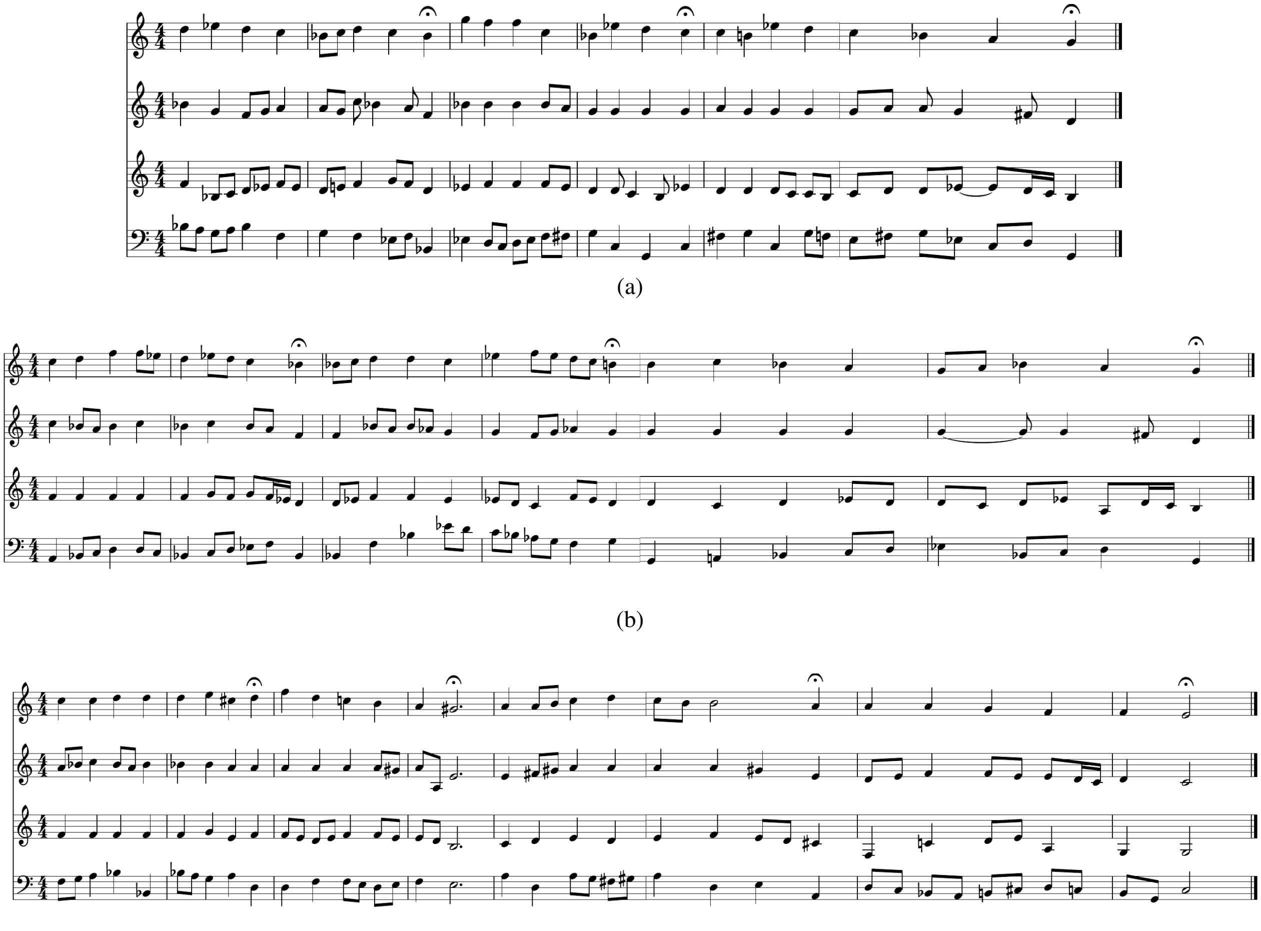
We described DeepBach, a probabilistic model together with a sampling method which is flexible, efficient and provides musically convincing results even to the ears of professionals. The strength of our method is the possibility to let users impose unary constraints, which is a feature often neglected 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 com-



Server address: http://localhost:5000/

deepbach

Figure 7. DeepBach's plugin minimal interface for the MuseScore music editor


(a)

(b)

(c)

Figure 8. Examples produced using DeepBach as an interactive composition tool. Examples (a) and (b) share the same metadata.

positional 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.

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

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Allan, Moray and Williams, Christopher KI. Harmonising chorales by probabilistic inference. *Advances in neural information processing systems*, 17:25–32, 2005.

Bach, J.S. *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`.

Boulanger-Lewandowski, Nicolas, Bengio, Yoshua, and Vincent, Pascal. 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)*, pp. 1159–1166, 2012.

Chen, Shyh-Huei and Ip, Edward H. Behaviour of the gibbs sampler when conditional distributions are potentially incompatible. *Journal of Statistical Computation and Simulation*, 85(16):3266–3275, 2015. doi: 10.1080/00949655.2014.968159. URL `http://dx.doi.org/10.1080/00949655.2014.968159`.

Chollet, François. Keras. `https://github.com/fchollet/keras`, 2015.

Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, and Bengio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Cuthbert, Michael Scott and Ariza, Christopher. music21: A toolkit for computer-aided musicology and symbolic music data. 2010.

De Sa, Christopher, Olukotun, Kunle, and Ré, Christopher. Ensuring rapid mixing and low bias for asynchronous gibbs sampling. *arXiv preprint arXiv:1602.07415*, 2016.

Ebcioglu, Kemal. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3):43–51, 1988. ISSN 01489267, 15315169. URL `http://www.jstor.org/stable/3680335`.

Geman, Stuart and Geman, Donald. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

Hadjeres, Gaëtan, Sakellariou, Jason, and Pachet, François. Style imitation and chord invention in polyphonic music with exponential families. *arXiv preprint arXiv:1609.05152*, 2016.

Heckerman, David, Chickering, David Maxwell, Meek, Christopher, Rounthwaite, Robert, and Kadie, Carl. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1(Oct):49–75, 2000.

Hild, Hermann, Feulner, Johannes, and Menzel, Wolfram. Harmonet: A neural net for harmonizing chorales in the style of js bach. In *Advances in neural information processing systems*, pp. 267–274, 1992.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kelly, Frank P. *Reversibility and stochastic networks*. Cambridge University Press, 2011.

Krauth, W. *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`.

Li, Xiangang and Wu, Xihong. 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)*, pp. 4520–4524. IEEE, 2015.

Liang, Feynman. Bachbot. `https://github.com/feynmanliang/bachbot`, 2016.

Lyu, Qi, Wu, Zhiyong, Zhu, Jun, and Meng, Helen. Modelling high-dimensional sequences with lstm-rtrbm: application to polyphonic music generation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pp. 4138–4139. AAAI Press, 2015.

Mikolov, Tomas, Joulin, Armand, Chopra, Sumit, Mathieu, Michael, and Ranzato, Marc'Aurelio. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*, 2014.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

Papadopoulos, Alexandre, Roy, Pierre, and Pachet, François. *Assisted Lead Sheet Composition Using Flow-Composer*, pp. 769–785. Springer International Publishing, Cham, 2016. 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.

Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. How to Construct Deep Recurrent Neural Networks. *ArXiv e-prints*, December 2013.

Sakellariou, J., Tria, F., Loreto, V., and Pachet, F. Maximum entropy model for melodic patterns. In *ICML Workshop on Constructive Machine Learning*, Paris (France), July 2015.

Sakellariou, J., Tria, F., Loreto, V., and Pachet, F. Maximum entropy models capture melodic styles. *ArXiv e-prints*, October 2016.

Vucelja, M. Lifting – A nonreversible Markov chain Monte Carlo Algorithm. *ArXiv e-prints*, December 2014.

Whorley, Raymond P. and Conklin, Darrell. Music generation from statistical models of harmony. *Journal of New Music Research*, 45(2):160–183, 2016. doi: 10.1080/09298215.2016.1173708. URL http://dx.doi.org/10.1080/09298215.2016.1173708.

Whorley, Raymond P, Wiggins, Geraint A, Rhodes, Christophe, and Pearce, Marcus T. Multiple viewpoint systems: Time complexity and the construction of domains for complex musical viewpoints in the harmonization problem. *Journal of New Music Research*, 42(3): 237–266, 2013.