DeepBach: a Steerable Model for Bach chorales generation

Gaëtan Hadjeres[1,2], François Pachet[1,2]

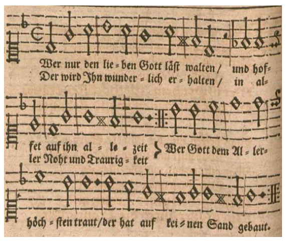
[1] Sony Computer Science Laboratories, Paris
[2] LIP6, Université Pierre et Marie Curie
`gaetan.hadjeres@etu-upmc.fr`
`pachetcsl@gmail.com`

Abstract. 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 art of Bach chorales composition involves combining four-part harmony with characteristic rhythmic patterns and typical melodic movements to produce musical phrases which begin, evolve and end (cadences) in a harmonious way. To our knowledge, no model so far was able to solve all these problems simultaneously using an agnostic machine-learning approach. This paper introduces DeepBach, a statistical model aimed at modeling polyphonic music and specifically four parts, 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. We evaluate how indistinguishable our generated chorales are from existing Bach chorales with a listening test. The results corroborate our claim. A key strength of DeepBach is that it is agnostic and flexible. Users can constrain the generation by imposing some notes, rhythms or cadences in the generated score. This allows users to reharmonize user-defined melodies. DeepBach's generation is fast, making it usable for interactive music composition applications. Several generation examples are provided and discussed from a musical point of view.

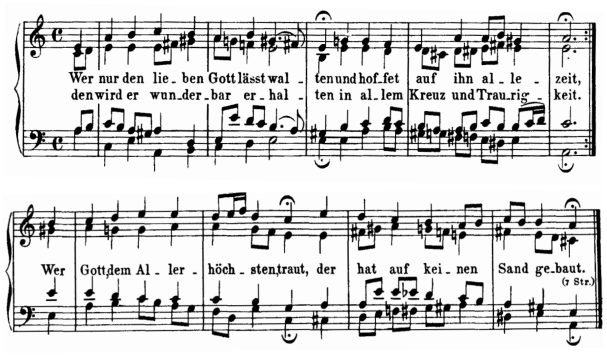
1 Introduction

The corpus of the chorale harmonizations by Johann Sebastian Bach is remarkable by its homogeneity and its size (389 chorales in [5]). 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. he composes the three lower parts (alto, tenor and bass) to be heard while the soprano (the highest part) sings the hymn, see Fig.1 for an example.

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 at the same time. This implies characteristic rhythms, variety of harmonic ideas as well as characteristic


Wer nur den lie-ben Gott läſt walten/ und hof-
Der wird Ihn wunder-lich er-halten/ in al-
fet auf ihn al-le-zeit
ler Noht und Traurig-keit
Wer Gott dem Al-ler-
höch-ſten traut/der hat auf kei-nen Sand gebaut.

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


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.
Wer Gott dem Al-ler-höch-sten traut, der hat auf kei-nen Sand ge-baut.
(7 Str.)

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

Fig. 1: Two versions of "Wer nur den lieben Gott läßt walten" original melody (a) and its reharmonization (b) by Johann Sebastian Bach (BWV 434) [3].

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 [13] 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. 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 arbitrary choices. Furthermore, we have no idea about the variety and originality of the proposed solutions.

A neural-network-based solution was later developed by [17]. 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 [3], but uses Hidden Markov Models (HMMs) instead

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

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 [29,28], 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 do 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, or the music by Bach) using neural networks have been investigated with promising results. In [8], chords are modeled with Restricted Boltzmann Machines (RBMs). Their temporal dependencies are learned using Recurrent Neural Networks (RNNs). Variations of these architectures have been developed, based on Long Short-Term Memory (LSTM) units [23] or GRUs (Gated Recurrent Units) [10]. These models, which work on piano roll representations of the music, are in our opinion too general to capture the specificity of Bach chorales. But one of their main 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.

The most recent advances in chorale harmonization is arguably the BachBot model [22], 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 and independently of our work, the authors evaluate their model with an online Turing test to assess the efficiency of their model with promising results. 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 LSTM-based model capable of producing musically-appealing four-part chorales in the style of Bach. Contrary to other models based on RNNs, we do not sample from left to right and model each voice separately. This allows us to enforce user-defined constraints such as rhythm, notes, parts, chords and cadences. DeepBach is able to produce coherent musical phrases and provides, for instance, varied reharmonizations of melodies without plagiarism. Its core features are its reliance upon no knowledge, its speed, the possible interaction with users and the richness of harmonic ideas it proposes. Its efficiency opens up new ways of creating interesting Bach-like chorales for non experts similarly to what is proposed in [26] for leadsheets.

In Sect. 2 we present the DeepBach architecture for four-part chorale generation.

We discuss in Sect. 3 the results of two experimental studies we conducted to assess the quality of our model. Finally, we provide several annotated examples in Sect. 4. All examples can be heard on the accompanying web page[4] and the code of our implementation is available on GitHub[5].

2 DeepBach

In this paper we introduce a new generative model which takes into account the distinction between voices. Sect. 2.1 indicates how we preprocessed the corpus of Bach chorale harmonizations and Sect. 2.2 presents the model's architecture.

2.1 Data Representation

We represent a chorale as a tuple of six lists

$$(\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{V}_4, \mathcal{S}, \mathcal{F}), \tag{1}$$

where the $\mathcal{V}_i$'s represent the four *voices* (soprano, alto, tenor and bass) to which we add two other lists: $\mathcal{S}$ the list of *subdivisions* and $\mathcal{F}$ the list of *fermatas*. All lists are indexed by a time index t and have equal size.

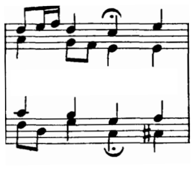
Since Bach chorales contains only simple time signatures, we discretize time with sixteenth notes, which means that each beat is subdivided into four equal parts. Since there is no smaller subdivision in Bach chorales, there is no loss of information in this process.

Each voice $\mathcal{V}_i$ contains the midi pitch of the played notes. It is a unique integer for each note, with no distinction between enharmonic equivalent notes. In order to represent rhythm in a compact way, we introduce an additional symbol to the pitches coding whether or not the preceding note is held. The *subdivision* list $\mathcal{S}$ 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. The *fermata* list $\mathcal{F}$ 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.

Fig. 2: A fermata symbol

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

[4] `http://www.flow-machines.com/deepbach-steerable-model-bach-chorales-generation/`
[5] `https://github.com/SonyCSL-Paris/DeepBach`



```
[74,__,76,77,74,__,__,__,72,__,__,__,76,__,__,__]
[69,__,__,__,67,__,65,__,64,__,__,__,64,__,__,__]
[60,__,__,__,59,__,__,__,55,__,__,__,57,__,__,__]
[53,__,50,__,55,__,__,__,48,__,__,__,49,__,__,__]
[ 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
[ 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0]
```

Fig. 3: Extract from a Bach chorale and its representation as six lists. The hold symbol is displayed as "__".

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. Part 4 shows that this representation makes our model able to create convincing musical phrases in triple and quadruple simple time signatures.

2.2 Model Architecture

For clarity, we suppose in this section that our dataset is composed of only one chorale written as in Eq. 1. We introduce a family of probabilistic models p parametrized by a parameter θ on our representation defined in Sect. 2.1. We do not model probabilistically the sequences $\mathcal{S}$ nor $\mathcal{F}$ but consider them fixed. The *negative log-likelihood* of our data is thus defined by

$$-\log p(\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{V}_4 | \mathcal{S}, \mathcal{F}, \theta). \tag{2}$$

We need to find a parameter θ which minimizes this loss. In order to have a computationally tractable training criterion, we introduce the *pseudolikelihood* of our data [7,4]. This approach was successful in many real-life problems [14] and consists in an approximation of the negative log-likelihood function by the sum over all variables:

$$-\sum_i \left(\sum_t \log p(\mathcal{V}_i^t | \mathcal{V}_{\setminus i,t}, \mathcal{S}, \mathcal{F}, \theta) \right), \tag{3}$$

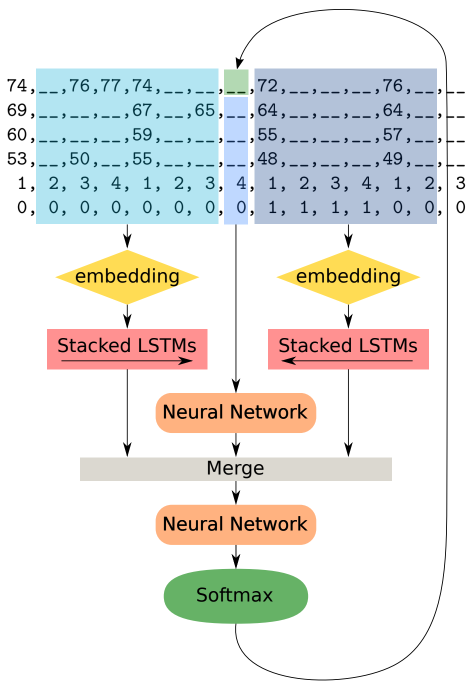
where $\mathcal{V}_i^t$ indicates the pitch of voice i at time index t and $\mathcal{V}_{\setminus i,t}$ the union of all $\mathcal{V}_i$'s except from the variable $\mathcal{V}_i^t$. This suggests to introduce four probabilistic models p_i depending on parameter θ_i, one for each voice, and to minimize their negative log-likelihood independently using the pseudolikelihood criterion. We obtain four problems of the form:

$$\sum_t \log p_i(\mathcal{V}_i^t | \mathcal{V}_{\setminus i,t}, \mathcal{S}, \mathcal{F}, \theta_i), \qquad \text{for } i \in [4]^6. \tag{4}$$

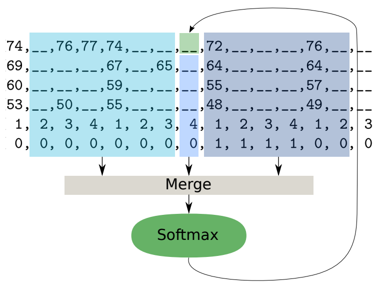
[6] We adopt the standard notation $[N]$ to denote the set of integers $\{1, \ldots, N\}$ for any integer N.

The advantage with this formulation is that each model has to make predictions within a small range of integer values whose ranges correspond to the usual voice ranges.

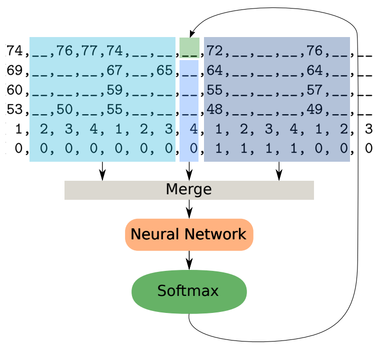
The aim of these models is to predict the pitch of one note knowing the value of its neighboring notes, the subdivision of the beat it is on and the presence of fermatas. We implement them using neural networks based on LSTMs [18,24]. For accurate predictions, we choose to use four neural networks: two stacks of LSTMs, 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. Their outputs are merged and passed as the input of a fourth neural network whose output is $p_i(\mathcal{V}_i^t|\mathcal{V}_{\setminus i,t}, \mathcal{S}, \mathcal{F}, \theta)$. Figure 4a shows a graphical representation for one of these models. Details are provided in Sect. 2.4.



(a) DeepBach



(b) MaxEnt



(c) MLP

Fig. 4: Graphical representations of neural networks for the soprano prediction p_1 for different models.

2.3 Generation

Generation is performed using Gibbs sampling [15]. In our case, this consists in the following algorithm:

Algorithm 1 Gibbs sampling

```
Require: Chorale length $L$, lists $\mathcal{S}$ and $\mathcal{F}$ of length $L$, probability distributions
    $(p_1, p_2, p_3, p_4)$, maximum number of iterations $M$

1: Create four lists $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{V}_4)$ of length $L$
   ▷ The lists are often initialized with random values drawn from the ranges of the
   corresponding voices
2: for $m$ from 1 to $M$ do
3:     Choose voice $i$ uniformly between 1 and 4
4:     Choose time $t$ uniformly between 1 and $L$
5:     Re-sample $\mathcal{V}_i^t$ from $p_i(\mathcal{V}_i^t|\mathcal{V}_{\setminus i,t}, \mathcal{S}, \mathcal{F}, \theta_i)$
6: end for
       return $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{V}_4)$
```

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

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

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

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. In Table 1 we show how the batch size (fixed number of parallel updates) influences the number of updates per second. Even if it known that this approach is biased [12] (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 [20]) to appropriately choose all the time indexes at which we parallelly resample so that:

Batch size	1	2	4	8	16	32	64	128	256	512	1024
Number of updates per second	59	112	200	334	508	636	693	824	914	974	1017

Table 1: Mean number of Gibbs updates per second during DeepBach's generation as a function of the batch size using a Nvidia GTX 980Ti GPU.

– 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.

2.4 Implementation Details

We implemented DeepBach using Keras [9] with the Tensorflow [1] backend. We used the database of chorale harmonizations by J.S. Bach included in the music21 [11] toolkit. 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 $(\mathcal{V}_{\setminus i,t}, \mathcal{S}\ , \mathcal{F})$ 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.

The reported results, Sect. 3, and examples Sect. 4 were obtained with $\Delta t =$ 16. We chose as the "neural network brick" in Fig. 4a a neural network with one hidden layer of size 200 and ReLU [25] nonlinearity and as the "Stacked LSTMs brick" two LSTMs on top of each other, each one being of size 200 (see Fig. 2 (f) in [21]). We experimentally found that adding dropout or sharing weights between the embedding layers improved neither validation accuracy nor the musical quality of our generated chorales.

3 Experimental Results

We now evaluate the quality of our model with two experiments: an online test conducted on human listeners and an analysis of the plagiarism in chorales generated by DeepBach.

3.1 Listening Test

The online listening test consists in a perception test and discrimination test. For the parameters used in our experiments, see Sect 2.4. We compared our model with two other models: a Maximum Entropy model (MaxEnt) as in [16] (Fig. 4b) and a Multilayer Perceptron (MLP) model (Fig. 4c).

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

$$p_i(\mathcal{V}_i^t|\mathcal{V}_{\setminus i,t}, \mathcal{S}, \mathcal{F}, A_i, b_i) = \text{Softmax}(AX + b) \tag{5}$$

where X is a vector containing the elements in $\mathcal{V}_{\setminus i,t} \cup \mathcal{S}_t \cup \mathcal{F}_t$, 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 pitches in the voice range i and Softmax the softmax function [30] 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 Multilayer Perceptron model we chose takes as input elements in $\mathcal{V}_{\setminus i,t} \cup \mathcal{S} \cup \mathcal{F}$, is a neural network with one hidden layer of size 500 and uses a ReLU [25] 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.

The musical extracts have been obtained by reharmonizing 50 chorales from the validation test by each of the three models (MaxEnt, MLP, DeepBach). We rendered the MIDI files using the Leeds Town Hall Organ soundfont[7] 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.

Perception Test In a first part, subjects were presented ten series of two reharmonizations of the *same* chorale melody and were asked "which one sounds more like Bach to your ears". In order to give a general ranking from these binary confrontations, we used the Bradley-Terry model [27,2] to infer potentials β_j

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

reflecting the probability that the version j is better than another version. This is expressed as:

$$P(\text{version } i \text{ is better than version } j) = \frac{e^{\beta_i}}{e^{\beta_i} + e^{\beta_j}}. \tag{6}$$

Results are plotted in Fig. 5.

1609 people took this test, 395 with musical expertise 1, 792 with musical expertise 2 and 422 with musical expertise 3.

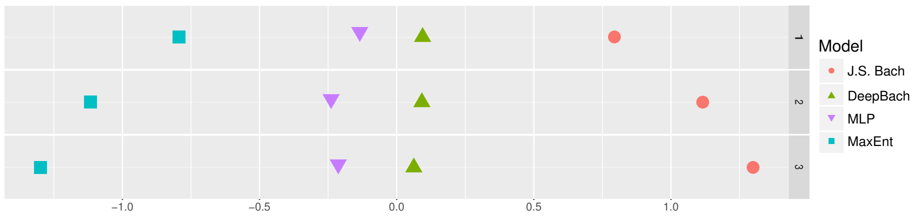


Fig. 5: Results of the perception test. The figure shows, for each level of expertise (from 1 to 3), the potentials of the Bradley-Terry model obtained from the pairwise comparisons. We centered the plots since only the distance between points matters. Better seen in color.

Extracts generated from DeepBach are clearly recognized as being more Bach-like than the other models. The more musical expertise subjects have, the clearer is the distinction.

Discrimination Test: "Bach or Computer" experiment In a second part, subjects were presented series of only one musical extract together with the binary choice "Bach" or "Computer"[8]. Fig. 6 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 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. 7 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

[8] This test is available at `http://www.flow-machines.com:3010`

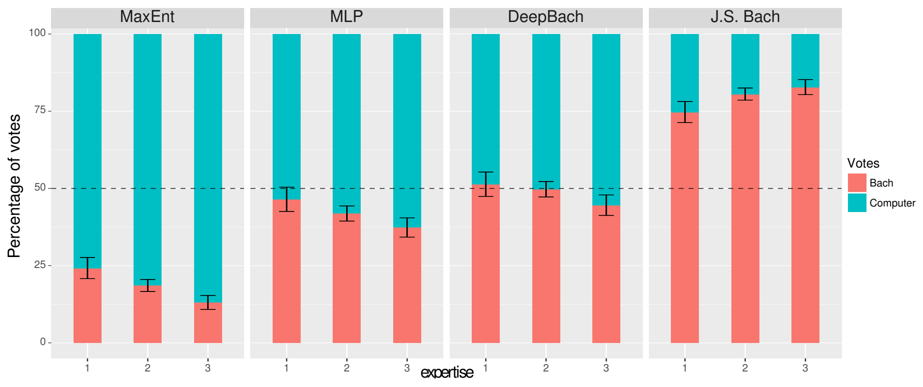


Fig. 6: Results of the "Bach or Computer" experiment. The figure shows the distribution of the votes between "Computer" (blue bars) and "Bach" (red bars) for each model and each level of expertise of the voters (from 1 to 3), see Sect. 3.1 for details.

votes considering them as being composed by J.S. Bach while it is only a third for the MLP model.

3.2 Plagiarism Analysis

We now evaluate the creativity and originality of DeepBach's productions. We use as a measure of plagiarism, for a given chorale, the length of the longest chorale subsequence which can be found identically in our training set. Fig. 8 shows histograms of this quantity for three different cases:

– 50 original (non transposed) J.S. Bach chorales from the test set
– 50 chorales (of the same length as the ones above) generated by DeepBach without constraints
– 50 reharmonizations by DeepBach where the soprano is constrained on the chorale melodies from the same 50 J.S. Bach chorales as above

For each case, we plot the length of the longest subsequence when considering only a given voice ("Soprano", "Alto", "Tenor" and "Bass" rows) and for all voices altogether ("All" row).

The results show that DeepBach is not prone to plagiarize both in the unconstrained generation and in the reharmonization cases. Indeed, when considering voices taken separately, we see that the distribution of the lengths of the longest plagiarized subsequence peaks around 5 or 6 beats. This can be compared with the distributions obtained on our J.S. Bach test set: chorales from this test set tend to be more "plagiaristic", with a higher mean length for the longest copied subsequences. This "self-plagiarism" is in fact characteristic of the style of the J.S. Bach chorales, with many typical movements, or cadences

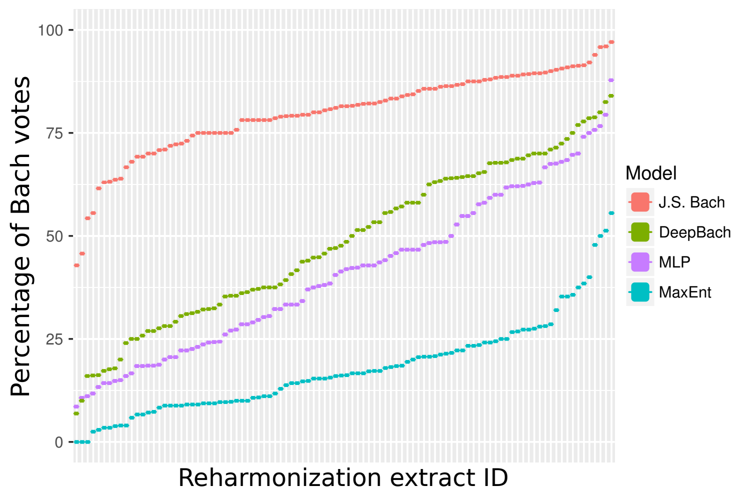


Fig. 7: Results of the “Bach or Computer” experiment. The figure shows the percentage of votes for Bach for each of the 100 extracts for each model. For each model, a specific order for the x-axis is chosen so that the percentage of Bach votes is an increasing function of the x variable, see Sect. 3.1 for details.

repeated exactly (up to transposition since we added all valid transpositions to our training dataset).

The peculiar histogram for the soprano voice for the J.S. Bach test set can be explained with two factors:

– the extreme values are due to the fact that J.S. Bach reharmonized some chorale melodies several times. This results in the presence of long copied subsequences.
– the central values are due to the particular combinatorics of the soprano voice. As chorale melodies are extracted from Lutheran hymns, their rhythm as well as the large use of step motions make them more prone to share common subsequences.

When looking to all voices simultaneously, we see that DeepBach does not suffer from plagiarism with copied sequences of small maximum size (around 2 beats). Even during reharmonization, we see that DeepBach is original enough so that only small subsequences (chord transitions) are cited verbatim in the generated chorales. This enables DeepBach to propose interesting and different reharmonization ideas of the same melody (see Fig. 10 and 11).

4 Commented examples

We now provide three complete chorale reharmonizations composed by DeepBach. One is a reharmonization of a chorale melody used by Bach (see Fig. 1)

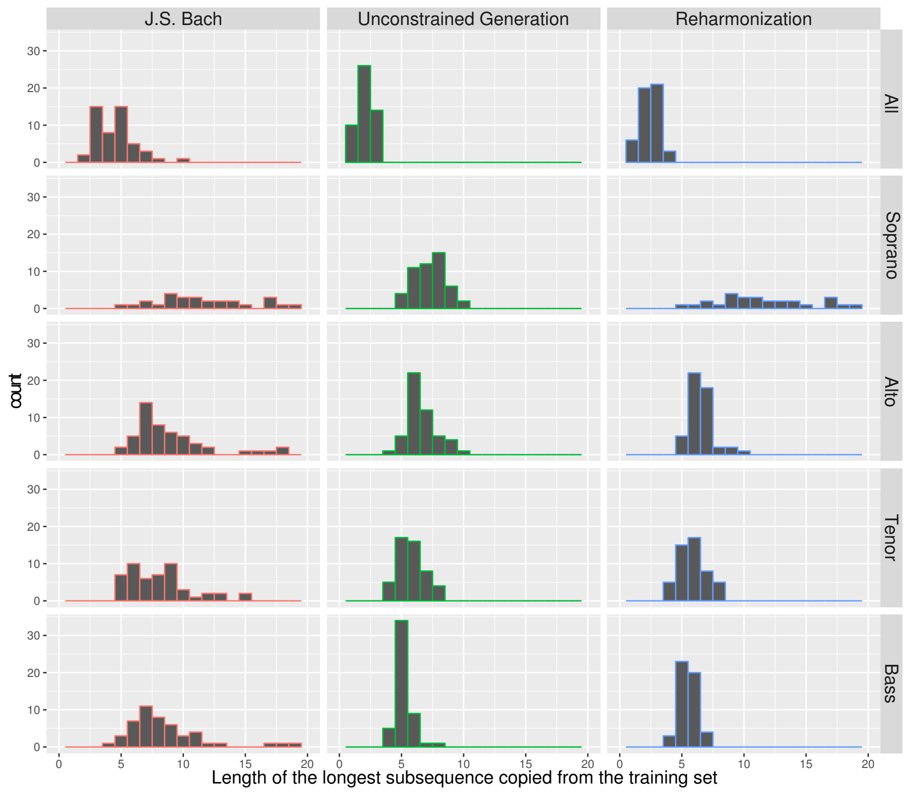


Fig. 8: Histograms of the length of the longest subsequence (in beats) copied from the training set for each voice in three different cases. See Sect. 3.2 for details.

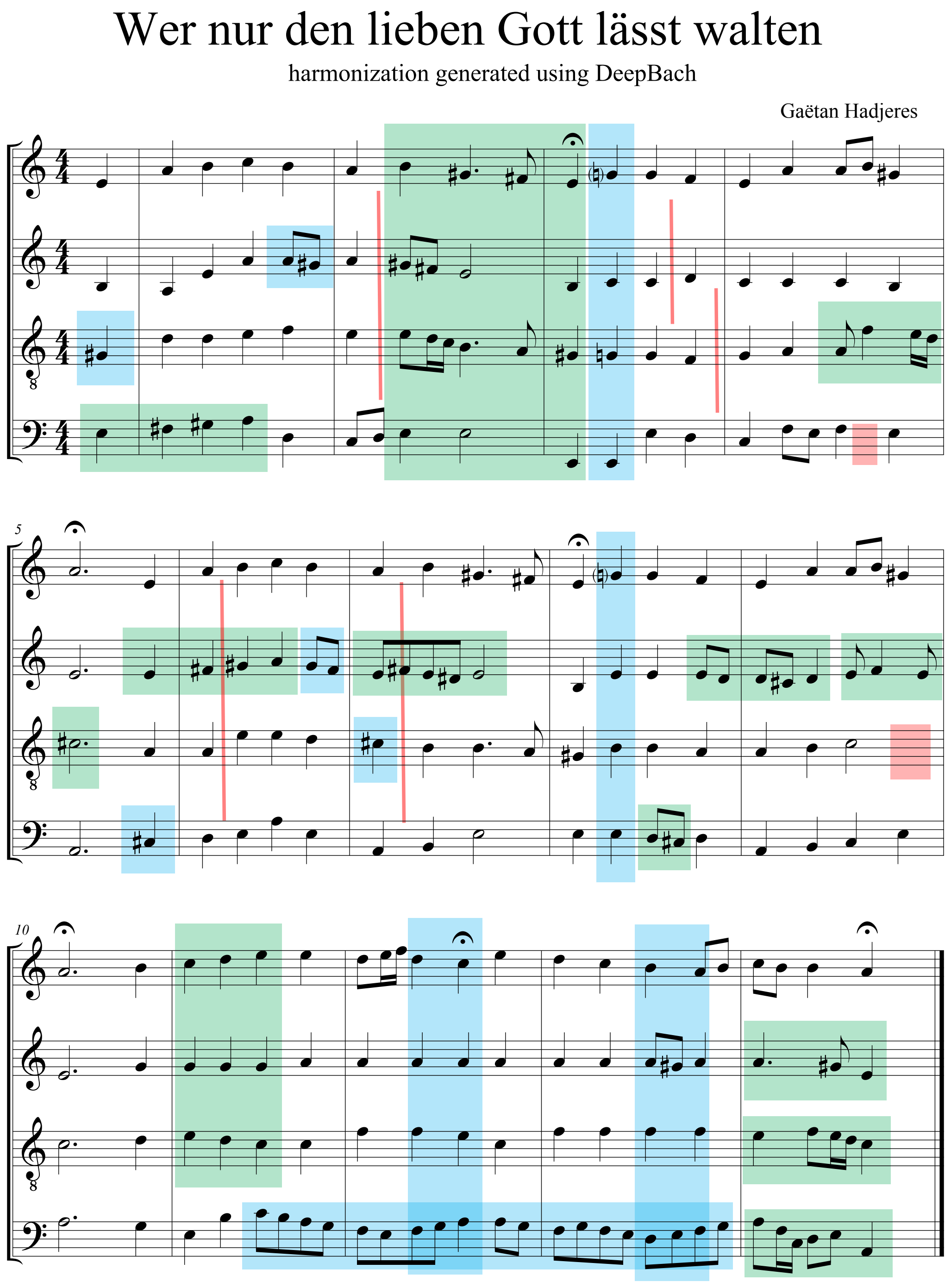

Wer nur den lieben Gott lässt walten
harmonization generated using DeepBach
Gaëtan Hadjeres
5
10

Fig. 9: Reharmonization of “Wer nur den lieben Gott läßt walten” by DeepBach. See Sect. 4 for comments on the annotations.

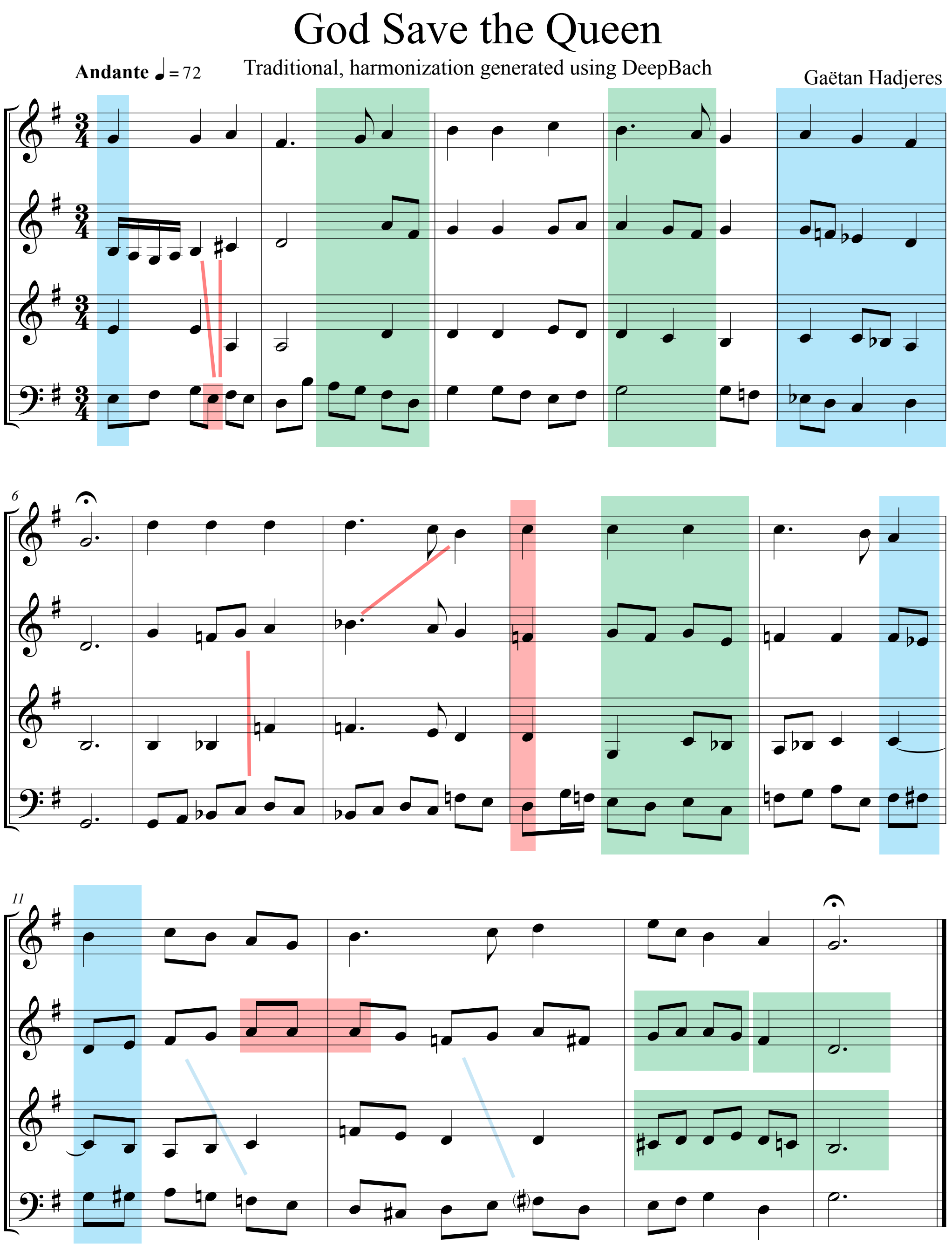


Fig. 10: A reharmonization of "God Save the Queen" by DeepBach. See Sect. 4 for comments on the annotations.

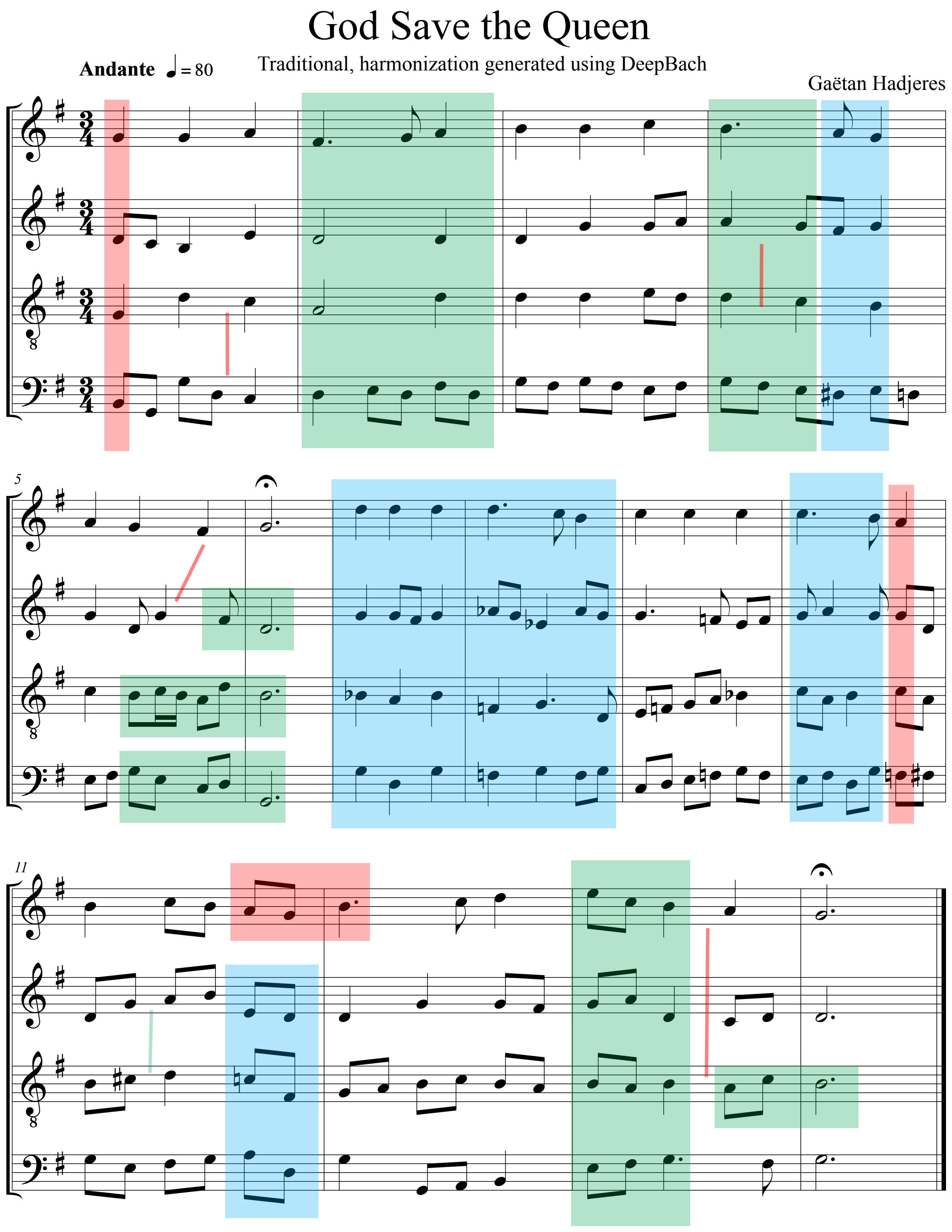


Fig. 11: A second reharmonization of “God Save the Queen” by DeepBach. See Sect. 4 for comments on the annotations.

while the other two are different reharmonizations of the traditional hymn "God Save the Queen"(see Fig. 10 and 11). 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 9 bars 1, 3, 6, 7, 9, 14
    - Fig 10 bars 13-14
    - Fig 11 bars 5, 13-14
  - good voicings and voice leading:
    - Fig 9 bars 2, 11
    - Fig 10 bars 2, 9
    - Fig 11 bars 2, 4, 13
  - characteristic suspensions[9] and passing tones:
    - Fig 9 bars 4, 8, 8-9, 14
    - Fig 10 bars 4, 13
    - Fig 11 bar 4
- in *blue*:
  - musically interesting ideas:
    - Fig 9:
      - 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 10:
      - 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 11:
      - 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 9:
      - D missing bar 4

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

    - C should resolve to B bar 9
  - Fig 10:
    - 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 11:
    - 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".

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, to our point of view, is the possibility to let users impose unary constraints, which is a feature often neglected in probabilistic models of music. We showed that DeepBach do not suffer from plagiarism while reproducing J.S. Bach's style and can be used to generate musically convincing harmonizations. We now plan to develop a music sheet graphical editor on top of the music21 toolkit in order to make interactive composition using DeepBach easier. This method is not only applicable to Bach chorales but embraces a wide range of polyphonic chorale music, from Palestrina to Take 6.

Acknowledgment

We thank Emmanuel Deruty for the audio rendering part, Jason Sakellariou for the fruitful discussions we had and Frank Nielsen for his helpful comments.

This research is conducted within the Flow Machines project which received funding from the European Research Council under the European Unions Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 291156.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `http://tensorflow.org/`, software available from tensorflow.org
2. Agresti, A., Kateri, M.: Categorical Data Analysis, pp. 206–208. Springer Berlin Heidelberg, Berlin, Heidelberg (2011), `http://dx.doi.org/10.1007/978-3-642-04898-2_161`
3. Allan, M., Williams, C.K.: Harmonising chorales by probabilistic inference. Advances in neural information processing systems 17, 25–32 (2005)
4. Arnold, B.C., Strauss, D.: Pseudolikelihood estimation: some examples. Sankhyā: The Indian Journal of Statistics, Series B pp. 233–243 (1991)
5. Bach, J.: 389 Chorales (Choral-Gesange): SATB (German Language Edition). Kalmus Classic Edition, Alfred Publishing Company (1985), `https://books.google.fr/books?id=U1-cAAAACAAJ`
6. Benward, B.: Music in Theory and Practice Volume 1. McGraw-Hill Higher Education (2014)
7. Besag, J.: Statistical analysis of non-lattice data. The statistician pp. 179–195 (1975)
8. Boulanger-lewandowski, N., Bengio, Y., Vincent, P.: 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)
9. Chollet, F.: Keras. `https://github.com/fchollet/keras` (2015)
10. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
11. Cuthbert, M.S., Ariza, C.: music21: A toolkit for computer-aided musicology and symbolic music data (2010)
12. De Sa, C., Olukotun, K., Ré, C.: Ensuring rapid mixing and low bias for asynchronous gibbs sampling. arXiv preprint arXiv:1602.07415 (2016)
13. Ebcioglu, K.: An expert system for harmonizing four-part chorales. Computer Music Journal 12(3), 43–51 (1988), `http://www.jstor.org/stable/3680335`
14. Ekeberg, M., Lövkvist, C., Lan, Y., Weigt, M., Aurell, E.: Improved contact prediction in proteins: using pseudolikelihoods to infer potts models. Physical Review E 87(1), 012707 (2013)
15. Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on pattern analysis and machine intelligence (6), 721–741 (1984)
16. Hadjeres, G., Sakellariou, J., Pachet, F.: Style imitation and chord invention in polyphonic music with exponential families. arXiv preprint arXiv:1609.05152 (2016)
17. Hild, H., Feulner, J., Menzel, W.: Harmonet: A neural net for harmonizing chorales in the style of js bach. In: Advances in neural information processing systems. pp. 267–274 (1992)

18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)
19. Koechlin, C.: Traité de l'harmonie: en 3 volumes, vol. 1. Eschig (1928)
20. Krauth, W.: Statistical Mechanics: Algorithms and Computations. Oxford Master Series in Physics, Oxford University Press, UK (2006), https://books.google.fr/books?id=EnabPPmmS4sC
21. Li, X., Wu, X.: 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)
22. Liang, F.: Bachbot. https://github.com/feynmanliang/bachbot (2016)
23. Lyu, Q., Wu, Z., Zhu, J., Meng, H.: 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)
24. Mikolov, T., Joulin, A., Chopra, S., Mathieu, M., Ranzato, M.: Learning longer memory in recurrent neural networks. arXiv preprint arXiv:1412.7753 (2014)
25. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp. 807–814 (2010)
26. Papadopoulos, A., Roy, P., Pachet, F.: Assisted Lead Sheet Composition Using FlowComposer, pp. 769–785. Springer International Publishing, Cham (2016), http://dx.doi.org/10.1007/978-3-319-44953-1_48
27. Ralph Allan Bradley, M.E.T.: Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika 39(3/4), 324–345 (1952), http://www.jstor.org/stable/2334029
28. Whorley, R.P., Conklin, D.: Music generation from statistical models of harmony. Journal of New Music Research 45(2), 160–183 (2016), http://dx.doi.org/10.1080/09298215.2016.1173708
29. Whorley, R.P., Wiggins, G.A., Rhodes, C., Pearce, M.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)
30. Wikipedia: Softmax function — Wikipedia, the free encyclopedia (2016), https://en.wikipedia.org/wiki/Softmax_function, [Online; accessed 7-november-2016]