Artificial Music Producer: Filtering Music Compositions by Artificial Taste

Fabian Ostermann, Igor Vatolkin, and Günter Rudolph
Department of Computer Science
TU Dortmund University, Germany
{fabian.ostermann;igor.vatolkin;guenter.rudolph}@udo.edu

Abstract

Human composers arrive at creative decisions on the basis of their individual musical taste. For automatic algorithmic composition, we propose to embrace that concept and encode taste as binary classification task. We identify and reconsider an implicit assumption: each and every result of a successful composing algorithm should be of great quality. In contrast, we formulate a general concept of composer-producer collaboration: an *artificial music producer* that filters 'good' and 'poor' results of an independent composer can improve musical quality without the need of refactoring composing strategies. That way, creative programming can be divided into independent subtasks, which allow for modular (multi-agent) system designs as well as productive team development. In a proof-of-concept experiment, we perform the discrimination of real Bach chorales from *fakes* generated by *DeepBach* using neural networks. This leads to an improvement of the overall results and provides possibilities to explain model behavior. Our concept can effortlessly be transferred to any pre-existing music generator.

1 Introduction

Individual taste is part of our everyday life. But taste is not exclusively important to consumers. It is equally important to artists creating objects that are to be liked. Taste provides an intuitive guideline to evaluate creations according to terms of aesthetics. A music composer sitting at the piano with pen and paper, or nowadays in front of a computer using music notation software, at bottom uses her/his taste to make creative decisions. But what if the composer is no human but a machine?

Including algorithms in the process of music composing is no novel approach. In the pre-computer era, algorithms were executed manually [27]. Many effortful attempts were made to automatically compose music. They form the research domain of *Algorithmic Composition* [12]. Such attempts were heavily criticized with equal commitment ever since. Results are commonly declared 'poor' compared to human composers [12, p.561]. But what are the main reasons?

An attempt at explanation: Great composers are expected to satisfy a minimum standard of artistic quality in each new composition. This minimum, however, is only measured subjectively by critics or listeners' response. What is often overlooked: even great composers fail during creative processes. If something turns out not 'good' enough by self-imposed standards, they rework it or even start anew from scratch. Accordingly, the published work is the *filtered* result of multiple attempts. If one succeeds in each and every attempt with the greatest of ease, we speak of a *musical genius* [19].

Algorithmic composition systems using modern Machine Learning (ML) techniques usually have no automated measure to self-evaluate results [5, Section 8.6], in contrast to traditional approaches [12], e.g., evolutionary computation [26]. We argue that the developers then optimize their algorithms towards the goal, that all their results become as 'good' as possible. Thereby, they *implicitly* implement "good taste" within the models. As a consequence, each and every composition is expected to become at least 'good'. If one developer would succeed in that, the system would instantly gain a kind of ingenious quality according to the reasoning above. Consequently, all composing algorithms would be designed to reach no less than perfection, which is not and obviously cannot be the case.

It is overambitious in comparison to the moderate accomplishments of state-of-the-art computer composition [12, p.561].

Assuming the number of 'good' results increases with the number of attempts, computers have significant advantages over humans: they are both fast and inexhaustible. The concept of the *Billion Song Dataset* [6] demonstrated how computers are able to produce tremendous amounts of music. The main problem is quality. Filtering out 'poor' results would instantly improve such algorithms.

In the following section, we argue that musical taste can be reduced to a binary classification task. That leads to our proposal in Sect. 3: the *artificial music producer*. It is derived from observations of human composer-producer collaboration. For further differentiation, we compare our approach to related research topics in Sect. 4. Section 5 demonstrates practical applicability: various Artificial Neural Networks (ANNs) are used to discriminate Bach chorales from algorithmically generated fakes. The results are then sorted by quality. Therefore, we formalize the quality measurement (*Bachness*) in Sect. 7, which allows for interpretation and explanation of model behavior (a major research interest, see [5, Section 6.17]). In Sect. 8, the *artificial producer* is used to filter an endless generation of chorales by *Bachness* and, thereby, improve the quality of *DeepBach*'s final output. Conclusions follow in Sect. 9.

2 Filtering Music by Taste: a Binary Classification Task

The great jazz composer Duke Ellington (1899–1974) is quoted as having said:

> "There are two kinds of music. Good music, and the other kind."

What at first makes a good laugh, actually speaks volumes about how serious artists reflect their creative processes. "Kinds of music" is a term associated with musicologists. Ellington misguides the reader that he is going to say something about musical genre, epochs, or styles. But those categories were made up by listeners, critics, and musicologists. Ellington emphasizes that those distinctions provide critical limitations for creators. Composers want to make something that is "good music" (at least in their eyes). Something others may appreciate and *like* to experience – the opposite of 'not good', 'bad', 'poor'.

Embracing this idea, defining a classification task is straightforward: Given a musical object M, its preprocessed feature vector $\vec{m}$ serves as input to a classification function T that models the musical taste of an individual i and outputs just one of the two classes *good* $\smile$ or *bad/poor* $\frown$:

$$T_i(\vec{m}) = y \qquad \text{where } y \in \{\smile, \frown\} \tag{1}$$

At first sight, this approach may seem naive. But, in our opinion, this simplification provides major advantages. General Adversarial Networks (GANs) demonstrate how this is successfully used in autonomous machine learning (details follow in Sect. 4). Our approach, however, is different since we can have humans involved in the development of composing strategies. This provides a way to preserve creativity (concerning originality) within the musical results. For example: learning $T(\vec{m})$ and using it to filter out 'poor' results improves existing *artificial composers* without modifying or rewriting code. Further, one common problem in the domain of computer art is addressed: promising projects are sometimes badly documented or important details are missing making reproduction impossible. Thus, older projects cannot be continued or adopted by others. However, results can be improved by our filtering approach, because no alteration of existing systems is needed.

Another proposal: Teams of creative co-workers can share workload. One uses her/his creative efforts to develop algorithms generating new and original ideas of great variety (accepting some 'poor' outcomes). Another developer deploys ML algorithms that identify desired characteristics within the results. The work of the *artificial critic* [30] is a recent example. It shows: creative teamwork can successfully be organized and accelerated in a sort of multi-agent scenario.

3 Concept of the *Artificial Music Producer*

There is a long-lasting trend to include more and more technical aids in artistic processes. They allow music composers to construct their creative musical ideas in less time. Artists are always appreciating such gain in productivity, e.g., to not be interrupted in periods of artistic flow. We will formally present our observations in this section in order to develop application scenarios.

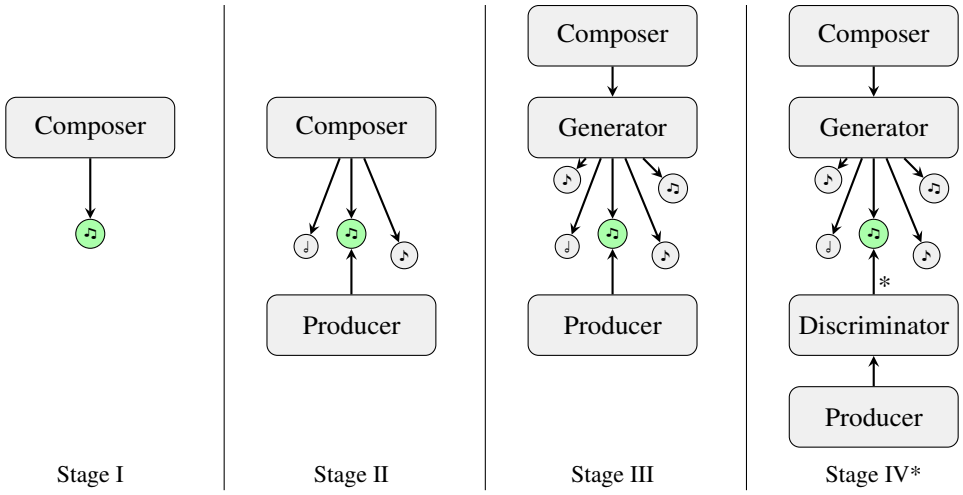


Figure 1: Evolution of the relationship of human music composers, human music producers and applied technical aids. Each circle represents one composition, the producer selects the 'best' (green).

The *artificial producer* is expected to classify compositions of known *artificial composers* in one of the classes $\smile$ or $\frown$. In the authors' opinion, this concept is the next logical step of an evolution (cf. Fig. 1) that already showed three stages I–III. The fourth stage IV* is our new proposal.

At first, the human composer is self-dependent by writing a piece and publishing it. In stage II, the composer provides a variety of pieces to a music producer, who as an experienced listener and professional critic chooses the 'best' creations. In the music industry, commercial reasons are driving forces [17]. But the concept can also be identified in other situations, like bands collectively deciding about songs to appear on an album, or one single composer looking back on former compositions.

Stage III is the state-of-the-art in modern music production. The *generator* may represent any kind of computer aid like music notation software featuring copy-and-paste, transposing and intelligent suggestions or digital audio workstations offering automatic pitch correction, software synthesizers and intelligent mixing templates. This way, composers can compose more pieces in less time, which results in additional workload for the producer. If the generator becomes a sophisticated composing algorithm, the time needed to create a piece is radically reduced. Naturally, no human can listen to music faster than it can be played. We propose to insert a discriminator (stage IV*), that automatically evaluates results by predicting producer decisions.

Note that our observations do not represent a one-way street. Today, some composers still work successfully by using just pen and paper, e.g., to intentionally boost their inner creativity by deliberate limitation. It means widening of possibilities and expanding the variety of how art takes shape. Further, we embrace artistic approaches. We think the terms generator and discriminator are too technical in a creative context. Hereinafter, we are using *artificial composer* and *artificial producer*, since we believe (just as [6]) that personification influences the way artistic tasks are being implemented in software.

4 Related Domains

The following research domains are related to our concept but hold significant differences.

General Adversarial Networks The concept from Fig. 1 has similarities to the principle of GANs [13], since they also use two ANNs named *generator* and *discriminator*. GANs, however, implement a fully automated version of our concept. The human composer and producer is the developer, who initially inputs example data, but acts fully passively afterwards. We find that thereby, originality is missing, since the GAN generator just imitates the statistical distribution of the example data. Our concept, instead, is open to include, e.g., knowledge-based systems or even human-in-the-loop procedures, which boost creative qualities of the results.

Recommender Systems Per definition, our concept is a *recommender system* (RS) as it makes predictions of feedback to consumable objects [20]. The main difference is modern RS practice, that has discarded content-based recommendation towards meta-information. *Collaborative filtering*, e.g., compares users' buying behaviors. For recommendation of music, cultural, demographic, political, and psychological background information have demonstrated applicability [22].

Hit Song Prediction This problem domain is a special case of RS. The task to predict the hits of past hit lists in order to predict hit potential of yet unreleased songs. Here, the *artificial producer* would not model the taste of one individual but of collective listeners. Motivation is provided by the music industry, that invests billions in finding new talents [17]. Main difference: only human-made popular music is considered. The goal of our concept is addressing and promoting algorithmic composition.

5 Design of Experiments

This section describes our proof-of-concept experiment. We address the question, whether a trained ANN functioning as *artificial producer* is able to filter undesirable output of the *artificial composer DeepBach* [15], that composes Bach-style chorales, and, thereby, improve the accepted output. We chose ANNs because of their flexible and adaptable environment that may simulate the complex cognitive processes of human music listening behavior [4, p.280]. We provide a comparative evaluation experiment in which we train and test different ANN types with various parameters and input representations to derive profound conclusions. T_i from Eq. (1) is approximated by supervised learning of 'good' and 'poor' examples.

5.1 Training Corpus of Real and *Fake* Bach Chorales

J. S. Bach left behind a collection of 389 four-part chorale harmonizations. They share similar characteristics making them popular for imitation exercises of music students and for equivalent ML tasks [2]. If the majority of composers and musicologists may agree on one thing, then that the work of Bach is "extraordinarily valuable". Therefore, we assign all chorales to $\cup$. Then *DeepBach* produces *fakes*, which are considered to be associated with $\cap$. The task of the *artificial producer* is then to learn to discriminate Bach chorales from fakes. Again, doing so may seem naive at first sight since it could imply that the detection of human vs. machine composed music is equivalent to detecting differences in taste. For the present experiment only, we defined to target Bach chorales as the ultimate goal, which will lead to interesting findings in the following. Our concept of the *artificial producer*, however, is not generally limited to a Turing-like human-or-machine distinction.

Similarities to GANs are obvious. The major difference is that we do not modify the working principle of *DeepBach* and, therefore, preserve the developer-intended behavior, which becomes crucial when dealing with more inventive generators as we plan for future work. Obviously, this is less important when dealing with full ML systems, but the implementation of *DeepBach* already cannot easily be transformed into a self-discriminating feedback loop [15, p.1365].

Prior to *DeepBach*, many attempts were made to algorithmically generate Bach-like chorales [1, 11, 18]. Unfortunately, none is working on modern hardware. Therefore, it is impossible to generate a training corpus. One exception is a dataset of 5000 Bach-style chorales [8] created with EMI [7] by David Cope. It was provided for exactly such research studies as ours, but unfortunately the server is down. The dataset seems lost and unreproducible for the time being.[1] Newer projects are *BachBot* [24] and *WaveGAN* [10]. But only *DeepBach* shares trained model parameters publicly, therefore it is chosen for our experiments.

The Bach chorales were retrieved as MIDI files from *music21* [9]. Omitting chorales not for SATB choir, 369 chorales remain. Hereinafter we refer to them as BACH_{369}. For each of the chorales a corresponding piece of the same bar length is created using *DeepBach*. This creates an analogous collection of 369 fake chorales, hereinafter referred to as FAKE_{369}. The union of $\text{BACH}_{369} \cup \text{FAKE}_{369}$ forms the training corpus.[2] A 5-fold cross-validation with training, validation and test subsets in ratio of 3:1:1 was applied to all experiments. Each chorale is assigned a subset as a whole. This results to $2 \cdot 369 \cdot \frac{3}{5} \approx 443$ chorales per training set.

[1] Correspondence with David Cope and former users led to this assessment.

[2] The datasets BACH_{369}, FAKE_{369}, and 1744 of the "perfect" chorales (see Sect. 8) are provided at: `https://doi.org/10.5281/zenodo.5726645`

5.2 Input Data Representations

Different symbolic and audio representations of the chorales are to be compared. First, MIDI events are transformed into human-readable text using `MIDICSV` [31]. Redundancies between repeating strings are removed by compressive encoding. Note that it can only be used with ANN types that can handle asynchronous input streams. Second, we arrange MIDI events as two-dimensional pitch-time diagram (Pianoroll, cf. Fig. 2) with a sampling rate of $100\,\mathrm{Hz}$. For audio, the MIDI files are synthesized using *The Leeds Town Hall Organ* samples kit [28]. This choice is consistent with the human evaluation in [15], so comparability is guaranteed. The audio sampling rate is $r{=}22.5\,\mathrm{kHz}$ mono. The third representation is a *Mel Spectrogram* (MelSpec) with windows of size $w{=}2048$ samples, 75% overlap and 128 Mel-bins. The fourth representation are 20 Mel Frequency Cepstral Coefficients (MFCCs).

5.3 Neural Network Types and Architectures

We also compare different ANN types. Fully-connected networks called *Multilayer Perceptrons* (MLPs), *Convolutional Neural Networks* (CNNs) and recurrent networks with *Long-Short Term Memory* (LSTM) units are our basic types (for details see [14]). For each type we compare different candidates with varying concrete architectures. Appendix A lists all hyperparameters.

The MLP is considered a baseline competitor. MLP1–3 have 1 to 3 fully-connected dense layers. MiniMLP is MLP1 with fewer weights in its hidden layers. DeepCNNdense is VGG16 [29] for image recognition with modifications to better match the audio domain [16]. Without final dense layer it becomes DeepCNN. Two shallow architectures named MiniCNN and MiniCNNdense are included for CNN baseline comparison. The recurrent candidates are a single LSTM unit with and without attached dense layer (LSTM and LSTMdense), two double-stacked LSTM units (LSTM2 and LSTM2dense, loosely following [23]) and a LSTM baseline competitor (MiniLSTM).

For training, we use *Adam* [21] with a learning rate of $\eta{=}0.001$ and early stopping (patience of 5 epochs, best weights restoring). The batch size is variable, because per batch we use all excerpts of one entire chorale to calculate the gradients. That way, each chorale has the same impact on the gradient descent regardless of its length.

6 Evaluation of Experiments

To find an optimal model for the task, all combinations of ANN candidates and representations were trained and tested with short excerpts of length $3\,\mathrm{s}$. This straightaway resulted in high accuracy values, so we decided to try larger excerpt lengths at a later point. Table 1 shows the excerpt-wise prediction accuracy (ACC) averaged over all cross-validation runs. Before discussing other statistical measures, we stick to ACC for initial evaluation, since we only treat binary classification with balanced training and test sets.

For our improved measure ACC*, we first average the excerpt predictions of each individual chorale. The prediction accuracy is then evaluated chorale-wise. Thereby, we interpret the output neurons' probability values as reliability measure. This correlation cannot generally be guaranteed. However, the fact that all ACC* values are greater than the corresponding ACC verifies our assumption within the scope of this experiment. Section 7 provide further insights and applications.

Table 1 holds interesting results: First, many combinations achieved a high ACC* even above 0.9. For Pianoroll, this is not unexpected. For MelSpecs, which masks symbolic features due to its overtone characteristics, the results were surprisingly good. Less surprising is the inferior performance of MFCCs, since their strength is providing information about timbre [25]. Nevertheless, the combination MiniCNN-MFCCs was able to achieve the highest ACC* of 0.951. Since all other models were not successful on MFCCs, we treat that result with caution.

MLPs did well on Pianoroll. They benefit from the consistent chorale form. CNNs, however, show an ambivalent picture. The deep versions, which are successful in recent applications, performed worse than MiniCNN. They are presumably too deep to learn the dataset's distribution. Attached dense layers also had negative impact. The LSTMs performed satisfactory. LSTM-Pianoroll and LSTM2dense-MelSpec were the best combinations. The attempts on MidiCSV succeeded and show potential, but could not be optimized further within the scope of this paper.

	ACC			ACC*			
	Pianoroll	MelSpec	MFCCs	Pianoroll	MelSpec	MFCCs	MidiCSV
MiniMLP	0.847	0.644	0.505	0.917	0.615	0.503	
MLP1	0.863	**0.659**	**0.515**	**0.935**	0.689	**0.552**	
MLP2	0.863	0.569	0.506	0.927	0.694	0.533	
MLP3	**0.866**	0.554	0.506	0.926	**0.699**	0.500	
MiniCNN	**0.848**	**0.838**	**0.819**	**0.946**	**0.942**	**0.951**	
MiniCNNdense	0.701	0.678	0.518	0.711	0.645	0.541	
DeepCNN	0.505	0.742	0.508	0.500	0.767	0.554	
DeepCNNdense	0.505	0.664	0.505	0.500	0.688	0.500	
MiniLSTM	0.823	0.821	0.504	0.882	0.901	0.500	**0.753**
LSTM	**0.825**	0.768	0.534	**0.905**	0.744	0.503	0.674
LSTM2	0.758	0.831	**0.629**	0.797	0.928	**0.550**	0.669
LSTMdense	0.812	0.814	0.534	0.867	0.919	0.500	0.748
LSTM2dense	0.699	**0.832**	0.543	0.680	**0.931**	0.500	0.500

Table 1: Initial comparison of the candidates by accuracy. Smaller numbers are grayed out. Highest values per column of each ANN type are in bold font.

	ACC					ACC*				
	1 s	3 s	5 s	10 s	12 s	1 s	3 s	5 s	10 s	12 s
MelSpec-MiniCNN	0.793	0.838	0.877	0.890	**0.892**	0.923	0.942	0.948	**0.950**	0.939
Pianoroll-MLP1	0.806	0.863	0.880	0.914	**0.918**	0.908	0.935	0.932	0.942	**0.943**
Pianoroll-MLP3	0.816	0.866	0.892	0.926	**0.927**	0.916	0.926	0.930	**0.957**	0.949
Pianoroll-MiniCNN	0.794	0.848	0.875	0.906	—	0.920	0.946	0.949	**0.961**	—
MFCCs-MiniCNN	0.756	0.819	0.859	0.888	**0.894**	0.928	0.951	**0.958**	0.943	0.951
human	—	0.630	0.750	0.740	0.690^{χ}					

Table 2: Comparison of the previously best combinations with increased input lengths. Value marked with χ is retrieved from [15, Fig.5].

In the next step, the excerpt length was optimized, cf. Table 2. It can be observed, that ACC* increases with larger excerpts and reaches its maximum at 10 s. In comparison with human assessment of 600 excerpts with three different excerpt lengths (by one of the authors), small excerpts of 3 s prove to be the harder task. However, 5 s and 10 s showed no difference. The value below 12 s is retrieved from the expert evaluation of *DeepBach* [15]. The lower value is explained by the fact, that the author gained more experience when preparing the training corpus. The subjects in [15] were experts, but they were unprepared for the specific task. Nevertheless, all models did indeed outperform the humans in predicting Bach or fake.

Final comparisons of the seven best performing models include precision P* and recall R* (cf. Table 3). We argue that P* is the most important value for our later example use case: since we will try to find the 'best' fake chorale by discarding 'poor' ones, we can rather accept 'good' chorales mistakenly removed (false-negative) than 'poor' ones mistakenly accepted (false-positive). Pianoroll-MLP3 with P*=1 did the *perfect* job on that.

7 The Concept of *Bachness* as a Quality Measure

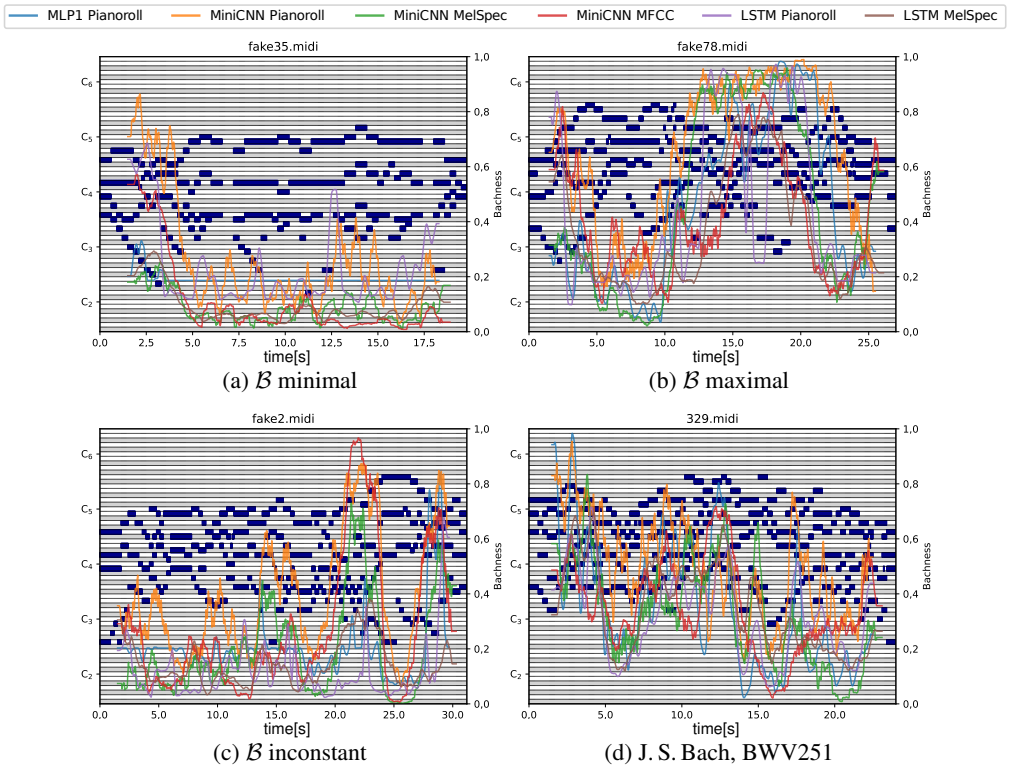
As we have seen, ACC* is significantly superior to ACC regarding correct classification. That proves that the output neuron of class $\smile$ carries information about the reliability of the classification. Therefore, we are able to not just filter results but to sort them according to their quality. Latter is expected to not only represent the experienced *correctness* of the chorales, but also a taste of magnificence in comparison to J. S. Bach, who was a great master in harmonization [3]. Therefore, the quality value was named *Bachness* factor $\mathcal{B}$.

	ACC*	P*	R*
MelSpec-MiniCNN (10 s)	0.950	0.991	0.908
Pianoroll-MLP1 (10 s)	0.942	0.994	0.889
Pianoroll-MLP1 (12 s)	0.943	0.986	0.900
Pianoroll-MLP3 (10 s)	0.957	**1.000**	0.913
Pianoroll-MiniCNN (10 s)	**0.961**	0.984	**0.938**
MFCCs-MiniCNN (10 s)	0.943	0.981	0.905
MFCCs-MiniCNN (5 s)	0.958	0.983	0.932

Table 3: Final comparison of the best models by chorale-wise accuracy, precision and recall.

Using $\mathcal{B}$ needs validation. The models are expected to generate similar values for the fake chorales. Therefore, a correlation analysis is applied to the top two models Pianoroll-MLP3 and Pianoroll-MiniCNN from Table 3. Average difference per chorale is $\overline{\Delta\mathcal{B}}$=0.038 with a standard deviation of $\sigma_{\Delta\mathcal{B}}$=0.036. The Pearson correlation coefficient confirms significant correlation with r=0.817. Even the ranks correlate with r=0.785. Therefore, the concept of *Bachness* is considered valid. We may then develop further analysis tools based on $\mathcal{B}$. For now, we propose *Bachness graphs* (Figure 2). They plot $\mathcal{B}$ over time visualizing changes of $\mathcal{B}$ within one chorale. The resulting $\mathcal{B}$-t diagrams are used to identify 'good' and 'poor' segments. All models show similar curves. That strengthens the argument of a general validity of $\mathcal{B}$ further.

Figure 2a shows the lowest rated chorale from FAKE_{369}. It shows massive tone repetitions creating stagnation. This is correctly detected because typical for Bach. Figure 2b shows the 'best' fake chorale. The *artificial producer* detects imperfections at the beginning. And indeed, e.g., around t=4 s there is an A major chord $\{C_4^\sharp, A_4, E_5\}$ on top of the bass note $C_3^{(\natural)}$. The disharmonic minor ninth interval is noticeable even for inexperienced listeners, thus correctly detected. Figure 2c has average *Bachness*. It attracts attention because of its fluctuations. Heavy harmonic movement is accredited high *Bachness* while long notes are penalized. Comparisons confirm that preference of Bach. Fig. 2d



(a) $\mathcal{B}$ minimal (b) $\mathcal{B}$ maximal

(c) $\mathcal{B}$ inconstant (d) J. S. Bach, BWV251

Figure 2: *Bachness* graph of the six best models, Pianorolls in backgrounds.

shows the 'poorest' of Bach's chorales. Again, movement is rewarded. The upper voices of the G major chord around t=16 s are $\{G_4, B_4, D_5\}$ and bass note is B_2. This is a large interval of a tenth plus another octave. It cannot be said (of course) that this is incorrect or disharmonic, but such a large range is rare to find in Bach chorales and thus a special difficulty for the *artificial producers*. Despite those interesting findings, further theoretical investigations remain future work.

8 The Perfect Fake Chorale

In this application scenario, *DeepBach* creates chorales nonstop. Meanwhile, the *artificial producer* (Pianoroll-MLP3) sorts them by $\mathcal{B}$. Within the scope of this study 1744 new chorales were generated.[2] Their duration was defined as BACH_{369}'s average of 24 s. Further statistics can be found in Appendix B: Fig. 3 shows the distribution of all 1744 $\mathcal{B}$ values. The climbing gradient to the right shows, that 'good' compositions are less frequent than those of mediocre quality.

On first listening impression, the results sound indeed as 'good' as claimed by [15]. The voices of the 'best' chorale provide sort of a contrapuntal movement. The second best takes some harmonic risks that Bach would rather have avoided, but dissonances are always resolved. The Top 10 compositions are pleasing even though showing brief moments of clumsiness.

At first sight, the Bottom 10 also seem quite fair. This impression is caused by the predominant use of major and minor triads. On further examination, we found heavy repetitive chords, tone repetitions and parallel harmonic movement. Some chorales just sound wild (negatively speaking).

In conclusion, the *artificial producer* was capable of filtering out chorales that had severe construction problems. That said, it definitely succeeded in improving *DeepBach*'s results. Besides, we observed that our concept brought additional excitement to the task of listening to the sorted collection of fake chorales. That is a quality, that must not be underestimated, because it can boost motivation and inspiration of both researchers and artists.

9 Conclusions and Outlook

The aim of this paper was to formalize a concept of composer-producer collaboration. Musical taste was defined as binary classification task. The simplistic approach showed major benefit when applied in a proof-of-concept experiment: Bach chorales were implicitly determined as generally appreciated by composers and musicologists. The publicly available Bach-style chorale generator *DeepBach* was used as an *artificial composer*. Various types of ANNs and music data representations were evaluated on their performance in discriminating the real Bach chorales from *fakes*. It was shown that improvement of results by filtering is possible and raw symbolic as well as raw audio information is suitable to predict (subjective) quality. The concept of *Bachness* as a measure of quality was verified by reliability assumptions about the output neurons. *Bachness* was then used to interpret model predictions and explain single decisions on the musical material. In a conclusive application scenario, we showed that a well-trained ANN can function as an *artificial music producer* filtering the output of a music generator.

Future work will include the application on other generators. Presumably, conducting a survey with human test subjects will be necessary to gain data on musical taste of different individuals. It will be of particular interest to investigate, if one single architecture is able to predict multiple human tastes, because it is likely that taste decisions base on dissimilar musical features. This will widen the application range of ML classification methods beyond objective tasks towards creative applications, e.g., controlling result characteristics (a current research topic, see [5, Section 6.10]).

Further, novel algorithmic composition systems should use a self-reporting *artificial producer* in different steps of their generation procedure (e.g., as successfully demonstrated by the artificial critic from [30]). An *artificial composer* may then more fully simulate the process of creativity in human artistic creation. This may also have reverse impact on the way developers think about their systems. In future, developing multi-agent systems with independent composer and producer units is a promising approach (as equally discussed in [5, Section 8.4]). Besides, we proposed our concept inside the domain of algorithmic music composition, but of course Fig. 1 can be effortlessly adopted to match other problem domains, e.g., *artificial painters*, *sculptures*, *writers*, *architects*, *fashion designers*, etc.

References

[1] M. Allan and C. K. I. Williams. Harmonising chorales by probabilistic inference. In *17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 25–32, Cambridge, MA, USA, 2004. MIT Press.

[2] C. Ames and M. Domino. Cybernetic composer: An overview. In *Understanding Music with AI: Perspectives on Music Cognition*, pages 186–205. MIT Press, Cambridge, USA, 1992.

[3] J. S. Bach. Die Kunst der Fuge. In *Bach Werke Verzeichnis, BWV 1080*. 1751.

[4] J. Berger. Who cares if it listens? An essay on creativity, expectations, and computational modeling of listening to music. In D. Cope, editor, *Virtual Music – Computer Synthesis of Musical Style*. MIT Press, Cambridge, USA, 2001.

[5] J.-P. Briot, G. Hadjeres, and F.-D. Pachet. *Deep Learning Techniques for Music Generation*. Springer, 2020.

[6] N. Collins. "...there is no reason why it should ever stop": Large-scale algorithmic composition. *Journal of Creative Music Systems*, 3(1), 2018.

[7] D. Cope. *Experiments in Musical Intelligence*. A-R Editions, Middleton, WI, USA, 1996.

[8] D. Cope. 5000 works in bach style, 2012. URL `http://artsites.ucsc.edu/faculty/cope/5000.html`. *Link to former project website: Download is currently unavailable.* (accessed: 25/07/2022).

[9] M. S. Cuthbert and C. Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In J. S. Downie and R. C. Veltkamp, editors, *11th International Society for Music Information Retrieval Conference*, pages 637–642, 2010.

[10] C. Donahue, J. McAuley, and M. Puckette. Adversarial audio synthesis. In *ICLR*, 2019.

[11] K. Ebcioğlu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3):43–51, 1988.

[12] J. Fernandez and F. Vico. AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48:513–582, Nov 2013.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[14] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL `http://www.deeplearningbook.org`. (accessed: 25/07/2022).

[15] G. Hadjeres, F. Pachet, and F. Nielsen. DeepBach: A steerable model for Bach chorales generation. In D. Precup and Y. W. Teh, editors, *34th International Conference on Machine Learning*, volume 70, pages 1362–1371, 2017.

[16] Y. Han, J. Kim, and K. Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):208–221, 2017.

[17] D. Herremans, D. Martens, and K. Sörensen. Dance hit song prediction. *Journal of New Music Research*, 43(3):291–302, 2014.

[18] H. Hild, J. Feulner, and D. Menzel. HARMONET: A neural net for harmonizing chorals in the style of J. S. Bach. *Advances in Neural Information Processing*, 4, 1992.

[19] B. L. Jacob. Algorithmic composition as a model of creativity. *Organised Sound: Special Issue on Algorithmic Composition*, 1(3):157–165, 1996.

[20] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2010.

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*, San Diego, USA, 2015.

[22] A. Laplante. Improving music recommender systems: What can we learn from research on music tastes? In *15th International Society for Music Information Retrieval Conference*, 2014.

[23] X. Li and X. Wu. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4520–4524. IEEE, 2015.

[24] F. Liang, M. Gotham, M. Johnson, and J. Shotton. Automatic stylistic composition of Bach chorales with deep LSTM. In *18th International Society for Music Information Retrieval Conference*, 2017.

[25] B. Logan. Mel frequency cepstral coefficients for music modeling. In *1st International Symposium Music Information Retrieval*, 2000.

[26] E. R. Miranda and J. A. Biles, editors. *Evolutionary Computer Music*. Springer, London, UK, 2007.

[27] W. A. Mozart. Anleitung zum Componieren von Walzern vermittels zweier Würfel. In *Köchelverzeichnis, KV 294d/516f*. Breitkopf & Härtel, Wiesbaden, 1862/1965.

[28] Samplephonics Ltd. *The Leeds Town Hall Organ*. Leeds, UK, 2014. URL `https://www.samplephonics.com/products/free/sampler-instruments/the-leeds-town-hall-organ`. (accessed: 25/07/2022).

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations*, 2015.

[30] B. Sturm. An artificial critic of irish double jigs. In *2nd Joint Conference on AI Music Creativity*, Online, 2021.

[31] J. Walker. *`MIDICSV` – translate MIDI file to CSV*. 2004. Version 1.1, 2008. URL `https://www.fourmilab.ch/webtools/midicsv/`. (accessed: 25/07/2022).

A Exact parameters of the ANN models

All hidden layers are using the ReLU activation function.
Output neurons are using softmax, except LSTMs are using sigmoid.

type	param.
Flatten	
Dense	32
Dropout	0.5
Dense	2

(a) MiniMLP

type	param.
Flatten	
Dense	256
Dropout	0.5
Dense	2

(b) MLP1

type	param.
Flatten	
Dense	256
Dropout	0.5
Dense	128
Dropout	0.5
Dense	2

(c) MLP2

type	param.
Flatten	
Dense	256
Dropout	0.5
Dense	128
Dropout	0.5
Dense	64
Dropout	0.5
Dense	2

(d) MLP3

Table 4: MLP candidates.

type	param.
Conv2D	16 Kernel (3×3)
Conv2D	16 Kernel (3×3)
Flatten	
[Dropout	0.5]
[Dense	128]
Dropout	0.5
Dense	2

(a) MiniCNN[dense]

type	param.
Conv2D	16 Kernel (3×3)
Conv2D	16 Kernel (3×3)
MaxPool	2×2 (Schritt 2)
Dropout	0.25
Conv2D	32 Kernel (3×3)
Conv2D	32 Kernel (3×3)
MaxPool	2×2 (Schritt 2)
Dropout	0.25
Conv2D	64 Kernel (3×3)
Conv2D	64 Kernel (3×3)
MaxPool	2×2 (Schritt 2)
Dropout	0.25
Conv2D	128 Kernel (3×3)
Conv2D	128 Kernel (3×3)
MaxPool	2×2 (Schritt 2)
Flatten	
[Dropout	0.5]
[Dense	128]
Dropout	0.5
Dense	2

(b) DeepCNN[dense]

Table 5: CNN candidates.

type	param.
Dropout	0.5
LSTM	128
Dropout	0.5
LSTM	32
Dense	2

(a) MiniLSTM

type	param.
Dropout	0.5
LSTM	256
[Dense	128]
Dense	2

(b) LSTM[dense]

type	param.
Dropout	0.5
LSTM	256
Dropout	0.5
LSTM	256
[Dense	128]
Dense	2

(c) LSTM2[dense]

Table 6: LSTM candidates.

B Statistics on the perfect fake chorale

id	rk	$\mathcal{B}$
01059	1	0.651
00237	2	0.604
00068	3	0.599
00123	4	0.593
00938	5	0.576
01331	6	0.533
01433	7	0.494
01378	8	0.489
00285	9	0.486
01297	10	0.484
⋮	⋮	⋮
00525	1724	0.006
00774	1725	0.006
0087	1726	0.006
00705	1727	0.006
0018	1728	0.005
0123	1729	0.004
00663	1730	0.004
0082	1731	0.003
00399	1732	0.003
01142	1733	0.002

Table 7: Top 10 and Bottom 10 list of chorales sorted by the *artificial producer* (Pianoroll-MLP3).

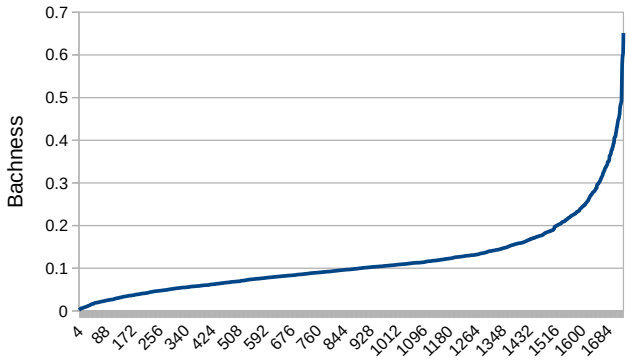


Figure 3: *Bachness* ($\mathcal{B}$) distribution of all 1744 fake chorales.