

This is a repository copy of *Measuring When a Music Generation Algorithm Copies Too Much: The Originality Report, Cardinality Score, and Symbolic Fingerprinting by Geometric Hashing*.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/190005/>

Version: Published Version

Article:

Yin, Zongyu, Reuben Paris, Federico orcid.org/0000-0003-1330-7346, Stepney, Susan orcid.org/0000-0003-3146-5401 et al. (1 more author) (2022) *Measuring When a Music Generation Algorithm Copies Too Much: The Originality Report, Cardinality Score, and Symbolic Fingerprinting by Geometric Hashing*. *SN Computer Science*. 340. ISSN 2661-8907

<https://doi.org/10.1007/s42979-022-01220-y>

Reuse

This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here:

<https://creativecommons.org/licenses/>

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.



Measuring When a Music Generation Algorithm Copies Too Much: The Originality Report, Cardinality Score, and Symbolic Fingerprinting by Geometric Hashing

Zongyu Yin¹ · Federico Reuben² · Susan Stepney¹ · Tom Collins²

Received: 11 September 2021 / Accepted: 19 May 2022
© The Author(s) 2022

Abstract

Research on automatic music generation lacks consideration of the originality of musical outputs, creating risks of plagiarism and/or copyright infringement. We present the *originality report*—a set of analyses that is parameterised by a “similarity score”—for measuring the extent to which an algorithm copies from the input music. First, we construct a baseline, to determine the extent to which human composers borrow from themselves and each other in some existing music corpus. Second, we apply a similar analysis to musical outputs of runs of MAIA Markov and Music Transformer generation algorithms, and compare the results to the baseline. Third, we investigate how originality varies as a function of Transformer’s training epoch. Fourth, we demonstrate the originality report with a different “similarity score” based on symbolic fingerprinting, encompassing music with more complex, expressive timing information. Results indicate that the originality of Transformer’s output is below the 95% confidence interval of the baseline. Musicological interpretation of the analyses shows that the Transformer model obtained via the conventional stopping criteria produces single-note repetition patterns, resulting in outputs of low quality and originality, while in later training epochs, the model tends to overfit, producing copies of excerpts of input pieces. Even with a larger data set, the same copying issues still exist. Thus, we recommend the originality report as a new means of evaluating algorithm training processes and outputs in future, and question the reported success of language-based deep learning models for music generation. Supporting materials (data sets and code) are available via <https://osf.io/96emr/>.

Keywords Music generation · Deep learning · Markov model · Originality evaluation

Introduction

A quotation from Igor Stravinsky reads: “A good composer does not imitate, he steals” [45]. The quotation, while made in relation to a serial work, reflects Stravinsky’s general interest in incorporating melodies, harmonic language, and forms from previous periods into new works such as his *Pulcinella Suite* (1922). Stravinsky uses the term “imitate” with a negative connotation: he would rather steal, say, a melody wholesale and rework it in a contemporary piece, than he would make mere allusions to (imitate) the work of past or contemporary composers. With respect to the current paper’s context—the rise of AI music generation algorithms—we instead use the term “imitate” with a positive connotation and the term “steal” with a negative connotation. As we show, some deep learning algorithms for music generation [19] are copying chunks of original input material in their output, and we would count it as a success if an algorithm—from the deep learning literature or otherwise—could

This article is part of the topical collection “Evolutionary Art and Music” guest edited by Aniko Ekart, Juan Romero and Tiago Martins.

✉ Tom Collins
tom.collins@york.ac.uk
<https://mstrcyork.org>

Zongyu Yin
zy728@york.ac.uk; zongyu.yin@outlook.com

Federico Reuben
federico.reuben@york.ac.uk

Susan Stepney
susan.stepney@york.ac.uk

¹ Department of Computer Science, University of York, York, UK

² Music, Science and Technology Research Cluster, Department of Music, University of York, York, UK

generate output that *sounds like* (imitates)—but does not *copy from* (steal)—pieces in a specific style.

Research on artificial intelligence (AI) has achieved various feats of simulating human perception (e.g., [17]) and production (e.g., [32]). A number of music generation models have been developed in recent decades, many predating or outside of deep learning [9, 38] and some espousing a belief in the superiority of deep learning [14, 33]. We have observed, with increasing alarm, that deep learning papers on music generation tend to rely solely or primarily on loss and accuracy as a means of evaluation [19, 33]. If there are listening studies, they employ listeners with inadequate expertise, and there is little or no musicological analysis of outputs, and no analysis of whether generated material plagiarises (steals from) the training data. As an increasing number of musicians are now incorporating AI into their creative workflows, checking an AI's output for plagiarism is now a paramount challenge in this area. To this end, this paper considers the topic of automatic stylistic composition—a branch of automatic music generation, where there is a stated stylistic aim with regards to the algorithm output, and a corpus of existing pieces in the target style.

In this context, we aim to establish a framework for checking the originality of auto-generated music with a specified style. We introduce and exemplify the originality report as a means of measuring when a music generation algorithm copies too much. We discuss how to calculate a distribution for the extent to which human composers borrow from themselves or each other in some corpus of pieces in a specific style; then we discuss how to use this as a baseline while moving a sliding window across a generated passage and measuring originality as a function of time in the generated material. The originality report is parameterised by a “similarity score”, so the framework is adaptable to measures that are more appropriate to certain characteristics of different data sets. We demonstrate the originality report with case studies of different similarity scores: the cardinality score and the fingerprinting score. The two measurement methods are used on different data sets, which differ mainly in terms of containing expressive timing data or not. The report is complemented by a musicological analysis of outputs from prominent deep [19] and non-deep [9] learning models. For the deep learning model, we also investigate how originality varies with training epoch.

Related Work

Music Plagiarism

Music plagiarism is said to have occurred when there is demonstrable and perceivable similarity between two songs or pieces of music (hereafter, pieces), and when there is

circumstantial evidence to indicate that the composer(s) of the latest piece would have been familiar with the existing piece. Stav [36] describes how the musical dimensions of melody, harmony, and rhythm contribute to music plagiarism, and gives an example-based explanation of how these dimensions have been used in handling music copyright disputes. Based on the features of melodies involved in selected plagiarism cases, Müllensiefen and Pendzich [26] derive an algorithm for predicting the associated court decision, and it identifies the correct outcome with 90% success rate. Recent failed or overturned cases also indicate that while music similarity and circumstantial evidence are necessary for delivering a verdict in favour of plagiarism having occurred, they are not sufficient, in that the distinctiveness of the music with respect to some larger corpus plays an important role too [6, 11, 28]: melodies that share contours and begin and end on the same scale steps may well point to potential cases of plagiarism, but it is likely that other melodies will have these same characteristics too [28]; drum beats, where the initial space of possibilities is smaller compared to pitched material, have been less successful as bases for music plagiarism convictions [30].

Recently, discussions on ethical issues surrounding AI have attracted widespread attention. Collins et al. [10] use a note-counting approach to show that twenty bars of computer-generated musical output from an algorithm by Cope [13] have 63% coincidence in pitch-rhythm combinations with a piece by Frédéric Chopin. In [37], a music generation algorithm's output and its tendency to copy original input pieces motivates the posing of open questions with respect to AI and music copyright law. As such generative models learn from existing music data, the copyright status of the output is unclear. In addition, the evaluation of these models' outputs tends to be narrow; it does not involve any kind of originality analysis with respect to the human-composed pieces used for training. This creates copyright or plagiarism infringement risks for musicians who are using these algorithms as part of their creative workflows.

Cognitive–Computational Approaches to Music Similarity

Largely outside of the role played by similarity in determining cases of music plagiarism, the systematic study of music similarity has a relatively long lineage [35] and continues to be of interest to scholars [41]. One challenging aspect of studying the phenomenon is that two excerpts of music can be similar to one another in myriad ways (genre, instrumentation, timbre, tempo, dynamics, texture, form, lyrics, and mentioned above, melody, harmony, rhythm). This challenge interacts with variability in use cases too. Take a single paradigm, such as query-based search in the form of music identification, which relies on some implementation of music

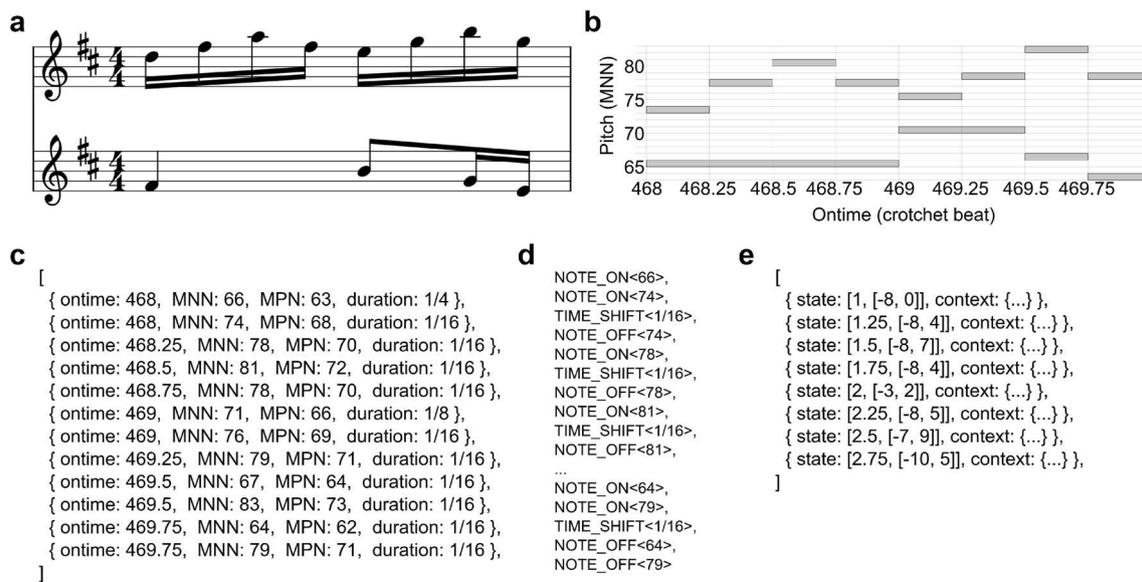


Fig. 1 Examples of symbolic music representations, starting from the same excerpt. **a** Half a bar of music; **b** the so-called piano-roll representation indicating some of the music's numeric properties; **c** a

four-dimensional representation of the music as a set of points; **d** one sequential representation that handles polyphony; **e** another sequential representation that handles polyphony

similarity. Even for this one paradigm, there are various use cases: Shazam addresses the need for exact matching [42], a variant of SoundHound addresses query-by-humming (the user sings or hums at the interface and expects “successful” results),¹ and Folk Tune Finder allows lyrics or notes to be input and, as with SoundHound’s query-by-humming variant, the user’s expectation of Folk Tune Finder is that the sought-after song will be found, or at least something relevant or interesting will be returned.² Of these use cases, only the one addressed by Shazam is clear cut—the other two are made more challenging by variation in cognitive and music-production capabilities of users, and there not necessarily being one “right answer”.

Here, we are concerned with a more reductive view of music similarity—the type of note/fingerprint-counting approaches mentioned above. This is the characterisation of music similarity that a teacher might employ if a student’s composition appears to draw too heavily on or copy directly from a known piece. For instance, “Why do 90% of the pitch-rhythm combinations in bars 1–20 of your piece occur also in this string quartet movement by Haydn?!” The representations and calculations required to reason this way, especially in algorithmic fashion, began in [22] and have been implemented in various forms since [1, 6, 39]. In the next section, we define two similarity measures based on the P3 algorithm [39] and the fingerprinting algorithm [1].

The fingerprinting approach in music shares some commonalities with image matching [23] and perceptual hashing [43, 46] in computer vision. Lowe’s [23] involves extracting distinctive features from images and using them to obtain the similarity to target images. This matching technique is shown to be robust to the affinity, noise, and change in viewpoint. Wang et al. [43] develop a perceptual image hash method, which generates hash codes based on image features to address the problem of content authentication (or, similarly, copyright infringement in images and videos). Zauner [46] proposes a benchmark framework for perceptual image hash functions, implemented in open-source software called pHash.³

We finish this section of the review with some remarks about choices of music representation and comparison methods. In general, researchers take sequential (e.g., [11, 12]) or geometric (e.g., [6, 25]) approaches to the representation and comparison of music. There are pros and cons to each approach. With the sequential approach, if one chooses to focus on MIDI note numbers (MNNs) alone and two melodies have the same MIDI notes (up to transposition) but different rhythms, a sequential representation (specifically, difference calculations between consecutive notes) will recognise these melodies as similar, whereas a geometric representation may not. However, with a sequential representation, it is less obvious how to handle polyphony (multiple notes beginning and ending at possibly different

¹ <https://www.midomi.com/>.

² <https://www.folktunefinder.com/>.

³ <https://www.phash.org/>.

times), whereas a geometric representation can encode a polyphonic piece as easily as it encodes a monophonic piece. For instance, in the sequential representation shown in Fig. 1d (which is Music Transformer’s [19] chosen input representation, see next section), the tokens encoding the occurrence of the F#4 and second F#5 are ten indices apart, even though the notes sound together. Therefore, any parameter that allows these events to be recognised as related has to be large enough to span this gap in indices. Moreover, an embellished (or, on the other hand, reduced) variation of some melody may not be recognised by the sequential representation as similar, because the relationships between adjacent notes will be altered by the added or removed notes, even though the “melodic scaffold” remains intact. A geometric representation may be more robust to this kind of variation.

Music Generation Models

Recently, a large number of deep learning models have been proposed for symbolic music generation [15, 19, 33]. Several of them regard music as a sequence of tokens, where generation involves predicting the next token based on previous tokens [19, 33]. Oore et al. [29] introduce a way to serialise polyphonic music and apply recurrent neural networks (RNNs) to generate output with expressive timing and velocity (loudness) levels. Huang et al. [19] use this same serialisation to adapt a transformer model [40] to generating music. Benefiting from the self-attention mechanism, it achieved lower validation loss compared to the RNN of [29] and also longer term stylistic consistency than previous RNNs-based approaches. In other work, based on the assumption that each musical output can be sampled from a normal distribution, [33] use variational autoencoders (VAEs) combined with long short-term memory networks (LSTMs). The application of generative adversarial networks (GANs) and convolutional neural networks (CNNs) to music generation has been explored also [15], using the piano-roll representation as in Fig. 1(b) and treating music as images that can be generated in a hierarchical manner.

An issue with all the above deep learning approaches to music generation is that there has been inadequate consideration of music plagiarism in the algorithms’ outputs. One user of the Music Transformer algorithm, Ruiz, writes:

The thing is that I ran the code on my machine and it overfits. It needs a way to check that it isn’t stealing from the data set say no more than 6 or 8 continuous notes. If it can’t do that it’s useless. I mean your piano data set is huge but after running the program for 20 times I found it composes note by note music of well known classical melodies. That’s not OK. That should be avoided [34].

Simon, a member of the Google Magenta team, replies:

In the checkpoints we’ve released, we tried hard to reduce the ability of the model to perform pieces from the train set. In addition, in the samples we released, we tried hard to remove any samples that are too similar to an existing piece of music. But it’s difficult to get to 100% on these for a number of reasons, including the lack of a clear definition for “too similar” [34].

Members of the general public can make use of Music Transformer for laudable reasons—Google Magenta have open-sourced the code—but their attempt to guard against music plagiarism appears problematic, and whatever constitutes “trying hard” in the above quotation has not been open-sourced, leaving general musicians who use Magenta algorithms in their creative workflows at risk of copyright infringement.

A non-deep learning approach to music generation that uses Markov models, pattern discovery, and pattern inheritance to ensure that generated material evidences long-term, hierarchical repetitive structure, also constitutes the first use of an originality or creativity analysis to assess the extent to which the model plagiarises human-composed works by Bach and Chopin on which it is based [9]. This algorithm, called MAIA Markov, uses the representation given in Fig. 1e, where each state consists of a beat of the bar and the MNNs relative to tonal centre occurring on that beat.

The remainder of this paper studies two of the most promising models for music generation, Music Transformer [19] and MAIA Markov [9], and focuses on the concept of originality, and methodologies for measuring it, which are then implemented and discussed.

Methods

This section introduces the method we use to analyse the originality of one set of symbolically encoded music excerpts relative to another. We begin by defining the two sets of excerpts: the queries (the excerpts we are testing for originality), \mathcal{Q} , and the targets (the excerpts we are testing against), \mathcal{R} . Depending on the use case, \mathcal{Q} may contain one or more excerpts from one or more pieces of music, and \mathcal{R} usually contains overlapping excerpts from multiple pieces of music. The use cases for wanting to produce an originality report, which we explore further and exemplify, are as follows:

1. The user wants to determine the “baseline” level of originality within a corpus \mathcal{C} . In this instance, the queries

\mathcal{Q} are a (pseudo-)random sample from \mathcal{C} , drawn from pieces \mathcal{Q}^* , and the targets \mathcal{R} are the set complement, $\mathcal{R} = \mathcal{C} \setminus \mathcal{Q}^*$.⁴ The outcome is a sample of N originality scores, from which estimates of the underlying distribution can be made, such as mean originality and confidence intervals about this mean.

2. The user wants to determine the level of originality of an algorithm’s output relative to a corpus whose contents the algorithm is designed to imitate. In this instance, the queries \mathcal{Q} would be overlapping excerpts of the algorithm’s output, and we plot the originality of elements of \mathcal{Q} as a function of time in the output, relative to elements of the target corpus \mathcal{R} . As well as plotting, we could also compare the mean or minimum originality found for the algorithm output to the distribution mentioned in the previous point, in which case the distribution acts as a “baseline” and can be used to address the question, “Is this algorithm’s output sufficiently original?”.
3. The user wants to incorporate originality reports into the modelling process itself, to analyse or steer/halt that process. The details are similar to the previous point, but the deployment of the method is during training or generation rather than after the fact.

Originality, Similarity, and Set of Points

To implement the originality reports that are associated with each of the use cases, it is necessary to employ at least one similarity measure—that is, some function $c : \mathcal{Q} \times \mathcal{R} \rightarrow [0, 1]$, which takes two symbolically encoded music excerpts q and r and returns a value in the range $[0, 1]$, indicating q and r are relatively similar (value near one) or dissimilar (value near zero). The measure ought to be commutative, $c(q, r) = c(r, q)$, and have a identity-like property that $c(q, q) = 1$. The choice of similarity measure influences subsequent decisions with respect to addressing questions, such as “Is this algorithm’s output sufficiently original?” Here we define the ideas of use cases and the originality report, and illustrate them in the context of two particular similarity measures: the cardinality score and the fingerprinting score.

Each originality report centres on calculating an originality score, OS , for some query q in relation to the set of targets \mathcal{R} . In particular, we find the element $r \in \mathcal{R}$ that maximises the similarity measure $c(q, r)$, and subtract it from 1:

$$OS(q, \mathcal{R}) = 1 - \max\{c(q, r) \mid r \in \mathcal{R}\} \tag{1}$$

So the originality score $OS : \mathcal{Q} \times \mathbb{P} \mathcal{R} \rightarrow [0, 1]$, where $\mathbb{P} \mathcal{R}$ is the power set of \mathcal{R} , is also a measure in the range $[0, 1]$. A value near one indicates query q is original relative to set \mathcal{R} ; a value near zero indicates it is unoriginal. If q is an exact copy of something that occurs in \mathcal{R} , then $OS(q, \mathcal{R}) = 0$.

Cardinality Score

One particular similarity measure is the cardinality score, cs [7, 20, 39]. Its strength is its simplicity and minimal parameter choices; its weakness is that if two sets of points (representing two note collections) differ from one another only by small but non-rigid amounts in one or more dimensions, cs will be near-zero, contrary to perception of the similarity between the two note collections still being high. The next section, on symbolic fingerprinting by geometric hashing, introduces an alternative similarity measure that addresses this weakness, but entails a larger number of parameter choices.

To calculate cs , we represent each music excerpt as a set of points containing the ontime⁵ (in crotchet beats) and numeric pitch representation (morphetic pitch number [24]) of each note.⁶ So an element of the query set $\mathcal{Q} : \mathbb{P}(N \times N)$ is represented as

$$q = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \tag{2}$$

and an element of the target set $\mathcal{R} : \mathbb{P}(N \times N)$ is represented as

$$r = \{(x'_1, y'_1), (x'_2, y'_2), \dots, (x'_n, y'_n)\} \tag{3}$$

An example of this representation is provided in Fig. 2c and d. The bottom-left point in (c) has the value $(x_1 = 468, y_1 = 53)$, representing an ontime at the beginning of the excerpt ($x_1 = 468$) and the morphetic pitch for C3 ($y_1 = 53$). The viola and cello have coincident notes at this moment, which project to a single point in our representation.

Letting t be the translation vector that gives rise to the maximum cardinality of the intersection $(q + t) \cap r$, we define the cardinality score as

$$cs(q, r) = |(q + t) \cap r| / \max\{|q|, |r|\} \tag{4}$$

where $|q|$ is the size of the set of points q . We demonstrate calculations of the cardinality score with reference to the

⁴ We distinguish between \mathcal{Q} and \mathcal{Q}^* , because if some excerpt $q \in \mathcal{Q}$ repeats or substantially recurs elsewhere in the piece \mathcal{Q}^* from which it is drawn, and we leave this repetition in the set of targets \mathcal{R} , then q would be considered trivially unoriginal. Therefore, it is sensible to hold out entire pieces from which queries are selected.

⁵ Ontime is the start time of a note counting in crotchet beats, with 0 for bar 1 beat 1 [5].

⁶ We use morphetic pitch in preference to MNNs here, because the former is robust to major/minor alterations.

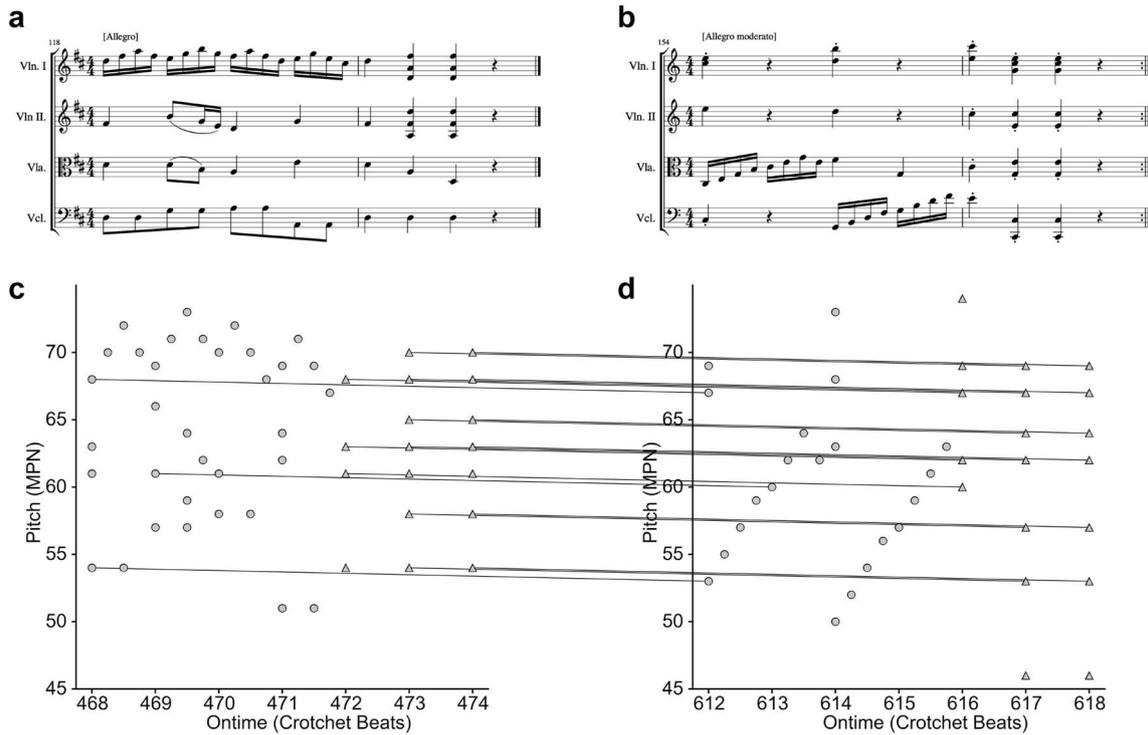


Fig. 2 Visualisation of the cardinality score between two excerpts. **a** A 2-bar excerpt from Mozart; **b** a 2-bar excerpt from Haydn; **c** mapping notes in the excerpt **a** to a set of points; **d** mapping notes

in the excerpt **b** to a set of points. For clarity, notes in the first/second bars are shown as circles/triangles

examples in Fig. 2. In the top half of this figure, there are two excerpts of string quartets: (a) is by Mozart and (b) is by Haydn. Considering the set of points corresponding to bars 119 of the Mozart and 155 of the Haydn (second bars in both Fig. 2a, b, with corresponding points shown as triangles), the vector $t = (-144, -2)$ translates 15 points from the Mozart excerpt to points in the Haydn, and the more numerous of the two sets is the Haydn excerpt, with 19 points, so the cardinality score is $cs(q, r) = 15/19 \approx 0.7895$. As a second example, considering larger point sets corresponding to bars 118–119 of the Mozart and 154–155 of the Haydn, the vector $t = (-144, -2)$ translates 18 points from the Mozart excerpt to points in the Haydn, and the more numerous of the two sets is the Mozart excerpt, with 52 points, so the cardinality score is $cs(q, r) = 18/52 \approx 0.3462$.

Symbolic Fingerprinting Using Geometric Hashing

Another approach for similarity measuring is symbolic fingerprinting [1, 6, 42]. Unlike the cardinality score above, where the similarity is measured notewise allowing for one rigid transformation vector corresponding to time and/or pitch shifts, the symbolic fingerprinting approach generates hash entries from triples of notes that satisfy various constraints of pitch and time differences, and then measures

the similarity via matching query entries to those in lookup table, when both are encountered at the corresponding start time (refer to hereafter the start timing counting in seconds). The selection of triples, hash construction, and matching process each confer robustness to non-rigid differences between two note collections, as we describe and demonstrate below, and which helps increase the perceptual validity of symbolic fingerprinting using geometric hashing as a similarity measure compared to cardinality score.

For geometric hashing, we again represent music excerpts as sets of points (see section “Cardinality Score”), and store the ratio (as similarity) of the number of matching hashes across a data set of pieces to the number of hashes generated by a query.⁷ Entries in the lookup table are constructed from target excerpts in advance. We first prepare each excerpt as a set of (start time, pitch) pairs (sorted by ascending start time), and then we generate hash entries of “valid triples” searched via Algorithm 1. A valid triple contains three pairs of start times and pitches that satisfy time/pitch-difference constraints. For example, the triple V is denoted as:

⁷ The implementation of fingerprinting algorithm is contributed by [8].

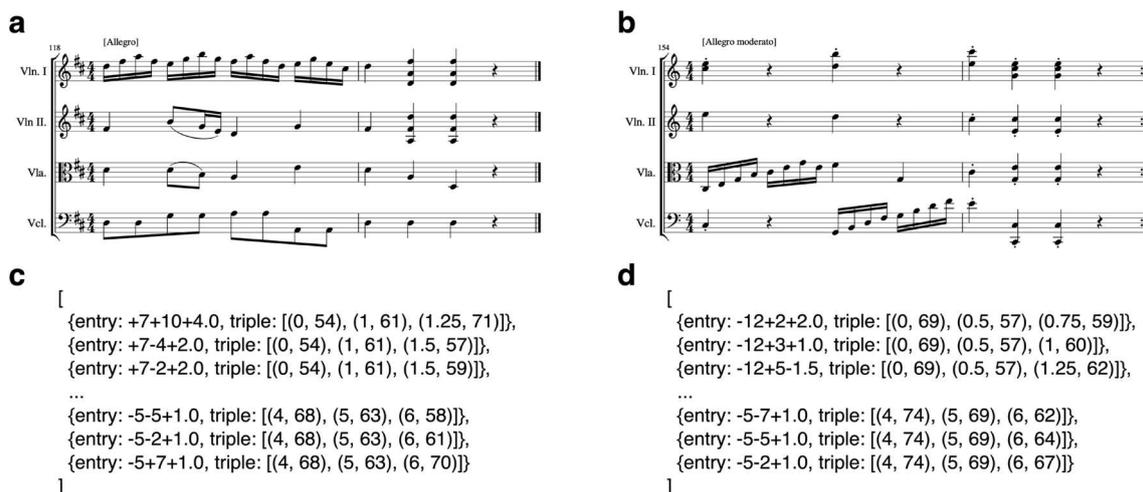


Fig. 3 Entries generated for two excerpts. **a** A 2-bar excerpt from Mozart; **b** a 2-bar excerpt from Haydn; **c** and **d** a list of entries generated following the corresponding triples of (start time, pitch)-pairs in the excerpt (**a**) and (**b**)

$$V = \{v_0, v_1, v_2\} \tag{5}$$

where each v has two attributes *start* and *pitch*. Regarding the pitch dimension, we obtain the pitch difference between two adjacent points, denoted as pd_0 and pd_1 :

$$pd_0 = v_1.pitch - v_0.pitch \tag{6}$$

$$pd_1 = v_2.pitch - v_1.pitch \tag{7}$$

We also calculate the intervals between start times, and obtain the ratio tdr instead of time difference, as tdr supports tempo-independent search (in other words, as per [22], the effect of scaling up/down the start time values can be mitigated or factored out entirely):

$$td_0 = v_1.start - v_0.start \tag{8}$$

$$td_1 = v_2.start - v_1.start \tag{9}$$

$$tdr = \begin{cases} td_0/td_1 & \text{if } td_0 \geq td_1 \\ td_1/td_0 & \text{if } td_1 > td_0 \end{cases} \tag{10}$$

We define a triple being valid as its pd_0 , pd_1 and tdr being in a certain range. We then encode pd_0 , pd_1 and tdr into a string as the hash entry, with the following steps:

1. For each of pd_0 and pd_1 , insert a + or - sign indicating a positive and negative relationship, and the absolute value of the pitch difference (e.g., +02, -10);

2. And then insert + if $td_0 \geq td_1$ and - if $td_1 > td_0$, preceding the float number of the tdr rounded to one decimal place.

For example, the entry string shown at the first item in Fig. 3c is +7 + 10 + 4.0, because the interval from D3 to D4 is +7 MPNs, and the interval from D4 to G5 is +10 MPNs; the time difference between the D3 and D4 is 1 crotchet beat, and the time difference between the D4 and G5 is 0.25 crotchet beats, which yields $tdr = +4.0$ as the time difference ratio.

Algorithm 1 consists of nested loops aiming to find valid triples, which are then encoded into an entry and inserted into a lookup table. The outer loop iterates through each point, denoted as v_0 . The nearest inner loop iterates through the points after v_0 , during which we apply the following constraints to find the second point v_1 : pitch difference between v_0 and v_1 , pd_0 , is in the range between $pMin$ (minimum pitch difference, e.g., 1) and $pMax$ (maximum pitch difference, e.g., 6); the start time difference between v_0 and v_1 , td_0 is in the range between $tMin$ (minimum time difference, e.g., 0.5) and $tMax$ (maximum time difference, e.g., 2). If the conditions fail, the loop skips to the next iteration. The most inner loop further iterates through points after v_1 , and then applies the same constraints to find v_2 to obtain pd_1 and td_1 .

Algorithm 1 Create entries from points

```

1:  $pts \leftarrow$  the set of points
2:  $npts \leftarrow pts.length$ 
3:  $tMin \leftarrow 0.5$ ;  $tMax \leftarrow 2$ ;  $pMin \leftarrow 1$ ;  $pMax \leftarrow 6$ 
4:  $lookup \leftarrow$  a hash table which maps entries to arrays of start times
5: for ( $i = 0$ ;  $i < npts - 2$ ;  $i++$ ) do
6:    $v_0 \leftarrow pts[i]$ 
7:    $j \leftarrow i + 1$ 
8:   while ( $j < npts$ ) do
9:      $v_1 \leftarrow pts[j]$ 
10:     $pd_0 \leftarrow abs(v_1.pitch - v_0.pitch)$ 
11:     $td_0 \leftarrow v_1.start - v_0.start$ 
12:    if ( $(td_0 > tMin) \wedge (td_0 < tMax) \wedge (pd_0 \geq pMin) \wedge (pd_0 \leq pMax)$ ) then
13:       $k \leftarrow j + 1$ 
14:      while ( $k < npts - 1$ ) do
15:         $v_2 \leftarrow pts[k]$ 
16:         $pd_1 \leftarrow abs(v_2.pitch - v_1.pitch)$ 
17:         $td_1 \leftarrow v_2.start - v_1.start$ 
18:        if ( $(td_1 > tMin) \wedge (td_1 < tMax) \wedge (pd_1 \geq pMin) \wedge (pd_1 \leq pMax)$ )
then
19:           $entry \leftarrow create\_entry(v_0, v_1, v_2)$ 
20:           $lookup.insert(entry, v_0.start)$ 
21:        end if
22:        if  $td_1 \geq tMax$ 
23:           $k \leftarrow npts - 1$ 
24:        end if
25:         $k++$ 
26:      end while
27:    end if
28:    if ( $td_0 \geq tMax$ ) then
29:       $j \leftarrow npts - 1$ 
30:    end if
31:     $j++$ 
32:  end while
33: end for

```

With a query q , we take the point-set representation and apply the same method as above to obtain entries. Each query entry is then used to match with the target entries saved in the lookup table, returning a point set $\{(a_1, b_1), (a_2, b_2), \dots, (a_M, b_M)\}$ of (target start time, query start time)-pairs. In a scatter plot of such data, true positives appear as (approximately) diagonal lines. Thus, we apply a simple transformation (e.g., $(a_i, b_i) \rightarrow (a_i, a_i - b_i)$) and calculate a histogram over this transformed data. Supposing there are k unique entries generated from a query, we calculate the similarity ratio to a certain target by dividing the number of unique matches h in a histogram bin by k .

Originality Reports

The data sets we use in this paper are:

- 71 MIDI encodings of Classical string quartet scores (sheet music) from the website KernScores.⁸ This data set was prepared according to the following filters and constraints:
 - string quartet composed by Haydn, Mozart, or Beethoven;
 - first movement;
 - fast tempo, e.g., one of Moderato, Allegretto, Allegro, Vivace, or Presto.
- 1276 MIDI encodings of performances on Yamaha Disklaviers by professional pianists of classical works from J.S. Bach to Alban Berg, from the MAESTRO data set version 3 [18].⁹ This data set is as follows:

⁸ See <https://osf.io/96emr/> for the data sets, algorithms, and analyses.

⁹ The difference between Classical and classical is meaningful. The large-“c” “Classical” period refers to Western art music composed in the period 1750–1830, whereas the small-“c” “classical” period subsumes this, comprising the period between 1650–1920.

- the annotated train set consists of 962 performances, the annotated validation set of 137 performances, and test set of 177 performances;
- if there is a performance of piece A in the train set, no other performances of piece A appears in the validation or test sets. Vice versa, if there is a performance of piece B in the validation or test set, no other performance of piece B appears in the train set.

The difference in the provenance of these two data sets (scores and performances) is important: a music score contains note start times and durations only up to some small integer subdivision of the main beat; a music performance captured via the MIDI format contains start times and durations in seconds (i.e., greater timing granularity).¹⁰ In other words, the MAESTRO data set contains the trace of human expressivity present in a performance, whereas the KernScores data set does not. It is not necessarily the case that having the expressive note start and duration timing data is advantageous, compared to the reduced set of values available in a score representation, however. Ideally, one would have access to both types of timing data, and the mapping between the two. The MAESTRO data set does not contain such a map, but we can still make use of it in this paper to address two questions:

1. How does the originality of output from the Music Transformer algorithm change as a function of data set, moving from a smaller data set to the larger one on which its original publication was based [19]?
2. Does the concept of the originality report stand up to scrutiny when we switch from a straightforward similarity measure that assumes score-like note start times (cardinality score, section “[Cardinality Score](#)”), to a more complex similarity measure that works also with expres-

sive note start times (geometric hashing, section “[Symbolic Fingerprinting Using Geometric Hashing](#)”)?

Sections “[Determining the Baseline Level of Originality Within a Corpus](#)”–“[Incorporating Originality Reports into an Algorithmic Process:Originality Decreases as Epoch Increases](#)” focus on originality reports obtained using cardinality score, and comparing performance of two algorithms in terms of the originality of their outputs. Then section “[Geometric Hashing: Producing Another Originality ReportWith a Different Similarity Measure, and Exploring AssociatedParameters](#)” provides analyses that address the two questions stated above.

Determining the Baseline Level of Originality Within a Corpus

To form the query and target sets for Classical string quartets data set, we divide the 71 excerpts into two sets: 50 queries \mathcal{Q} were drawn from 7 pieces \mathcal{Q}^* , and the targets \mathcal{R} consisted of the remaining 64 pieces. The selection of the 7 pieces was pseudo-random to reflect the representation of composers and time signatures in the overall data set. We use a fixed window size of 16 beats for each query.

To demonstrate the robustness of geometric hashing to music with expressive timing, the same approach is applied to the MAESTRO data set [18], using the 962 performances in the train set as the targets \mathcal{R} , and 50 queries \mathcal{Q} drawn from 314 performances \mathcal{Q}^* consisting of the amalgamation of the 137 validation and 177 test performances. We use a fixed window size of 8 s for each query.

We ran the code outlined in Algorithm 2 to obtain the baseline for both approaches using the corresponding similarity function c .

Algorithm 2 Estimating the self-originality of a corpus

Require: \mathcal{Q} , \mathcal{R} , query and target corpus, respectively.

```

Initialize  $O$  to an empty output list.
2: Initialize  $N$  to the number of originality scores.
   for ( $i := 0, i < N, i++$ ) do
4:    $q := \text{sample}(\mathcal{Q})$ 
   Initialize  $C$  as an empty list to store similarity scores.
6:   for each  $q \in \mathcal{Q}$  do
      $C.append(c(q, \mathcal{R}))$ 
8:   end for
      $O.append([1 - \max(C)])$ 
10: end for

```

¹⁰ There are also differences in dynamic level (loudness) and indications of which notes should be played by which instrument or hand, but they are not relevant to the current paper, so we will not elaborate on these further.

Fig. 4 Top 10 and last 10 hash entries that are generated as lookup table keys, with the descending order of occurrence count

Top 10

```
[
  { hash: +05-05+1.1, count: 268027 },
  { hash: +05-05-1.1, count: 264901 },
  { hash: -05+05+1.1, count: 255960 },
  { hash: -05+05-1.1, count: 255934 },
  { hash: +02-02+1.1, count: 254712 },
  { hash: +02-02-1.1, count: 253723 },
  { hash: -02+02+1.1, count: 248714 },
  { hash: -02+02-1.1, count: 246992 },
  { hash: -03+03+1.1, count: 237036 },
  { hash: -03+03-1.1, count: 235844 }
]
```

Last 10

```
[
  { hash: -02+01-4.0, count: 21 },
  { hash: -02+01+4.0, count: 21 },
  { hash: -01+02-4.0, count: 21 },
  { hash: +04-01-4.0, count: 21 },
  { hash: -06+04+4.0, count: 21 },
  { hash: -06-04+4.0, count: 21 },
  { hash: -02+06+4.0, count: 21 },
  { hash: -01-01-4.0, count: 20 },
  { hash: -04-06+4.0, count: 19 },
  { hash: +01+05-4.0, count: 19 }
]
```

For our sample of $N = 50$ excerpts from Haydn, Mozart, and Beethoven string quartets (KernScores), the mean originality is $\text{mean}(OS) = 0.699$, with bootstrap 95%-confidence interval 0.672 and 0.725. We interpret this to mean that for the current corpus and sample (space of fast, first-movement Classical string quartets), composers wrote music that is 69.9% original, at least according to the note-counting music similarity measure employed here. For the sample of $N = 50$ excerpts from classical piano performances (MAESTRO), the mean originality is $\text{mean}(OS) = 0.562$, with bootstrap 95%-confidence interval 0.546 and 0.579. As both the data set and similarity metric have changed, it is not meaningful to compare the two confidence intervals to one another. However, when music-generation algorithms are trained on one or the other corpus, it is meaningful to compare the originality of their outputs to their respective training corpus' self-originality, addressing the question of whether the music-generation algorithm's output is sufficiently original.

With the parameters $tMin \leftarrow 0.5$, $tMax \leftarrow 2$, $pMin \leftarrow 1$, $pMax \leftarrow 6$, the lookup table has a maximum 8, 928 unique hashes. This is because there are 31 possible tdr 's, 1.0, 1.1, ..., 4.0, as well as 2 possible signs for each (see Eq. 8). There are 2 pitch differences encoded in a hash (see Eq. 6), covering 6 possibilities 1, 2, ..., 6, again with 2 possible signs for each. This gives $8,928 = (31 \cdot 2) \cdot (6 \cdot 2)^2$ unique hashes in total. During the building of the lookup table across the MAESTRO train set, each of these possible hash codes was encountered, and we list the ten top and bottom ten entries ordered descending by frequency of occurrence in Fig. 4. From a musical point of view, the pitch content of the ten most common hashes are not surprising: they articulate rising and falling (or falling and rising) perfect fourths, (± 5 difference in MNNs), major seconds (± 2 difference in MNNs), and minor thirds (± 3 difference in MNNs). It has also been found previously that time difference ratios of close to 1 (e.g., ± 1.1) are more common than time difference ratios of exactly 1, suggesting that it is more desirable for pianists not to play such sequences absolutely isochronously, but to play one of the two time differences

slightly longer than the other [44]. As for the least common hashes, it is not surprising to find extremely imbalanced time-difference pairs (± 4.0) are rare, and include intervals of the tritone (± 6 difference in MNNs) and minor second (± 1 difference in MNNs).

Is This Algorithm's Output Sufficiently Original?

We can use the mean and confidence interval calculated above to help address the question of whether an algorithm's output is sufficiently original. Let us suppose we have a passage generated by an algorithm, and we traverse that output, collecting n -beat excerpts with 50% overlap, say, into a query set \mathcal{Q} . In this paper, we use $n = 8, 16$ beats, which corresponds to 2- and 4-bar excerpts in 4-4 time, respectively. It is advisable to use at least two different window sizes, to probe the assumption that originality should increase with window size. In other words, different window sizes can be used to determine whether a worrisome-looking instance of low originality at the 2-bar level increases—and so becomes less worrisome—at a longer 4-bar window size.

We ran the MAIA Markov [9] and Music Transformer algorithms [19] to explore this question of sufficient

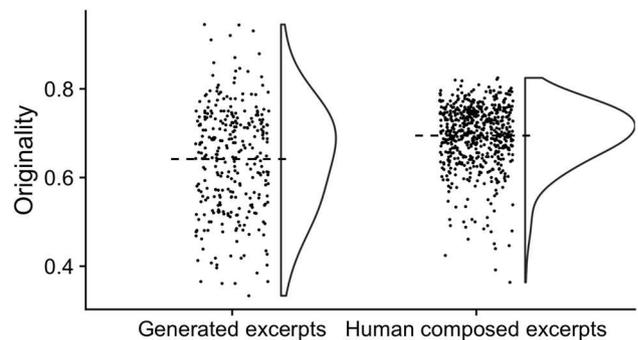


Fig. 5 Rain cloud plot of originality scores for both model generated excerpts and human composed excerpts, with the dashed lines showing mean values

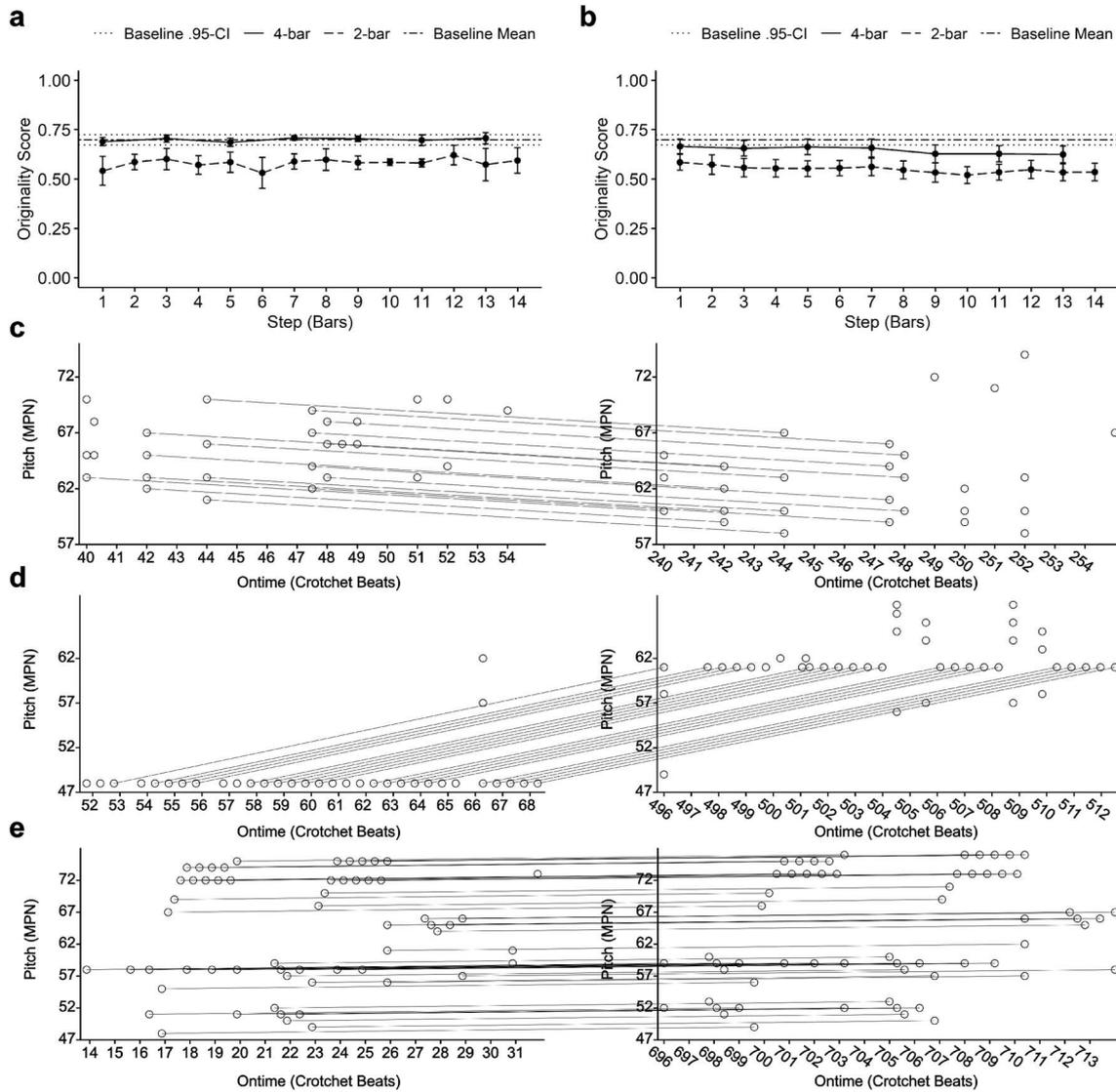


Fig. 6 Originality report for the MAIA Markov and Music Transformer algorithms. **a** and **b** Show the change in originality scores over the course of the excerpts obtained for MAIA Markov and Music Transformer, respectively, at 2- and 4-bar levels compared to

the baseline mean and 95%-confidence interval; **c–e** show worst-case examples of copying by MAIA Markov and Music Transformer at checkpoints 3 and 15, respectively, where the generated outputs are on the left and the human-composed excerpts are on the right

originality, based on the training data of 64 string quartet movements described above. Markov model was built on the representation shown in Fig. 1e. The Music Transformer model’s training data was augmented by transposing the original pieces in the range $[-5, 6]$ MIDI notes, and then we sliced these into subsequences of fixed size 2048 for batch training, giving a train set of 4128 and a validation data set of 564 subsequences. The model, with six layers, eight heads, and hidden size of 512, was trained with smoothed cross entropy loss [27] and the Adam optimiser [21] with custom learning rate schedule [3]. In keeping with the standard approach, the training process was stopped at epoch (checkpoint) 3, where the validation loss reached a minimum

value of 1.183. Afterwards, 30 excerpts were generated by MAIA Markov and Music Transformer to form the query set Q , based on which the mean value of originality scores were obtained by following the same method in section “Originality, Similarity, and Set of Points”, now for each time window as the excerpt is traversed.

Figure 5 aims to provide insight into how the originality scores distributed for both human composed excerpts and model generated excerpts, we obtained originality scores from 7 query pieces (human composed), and 30 pieces generated by the model at epoch 3 (the “best” model, see section “Incorporating Originality Reports into an Algorithmic Process:Originality Decreases as Epoch Increases”).

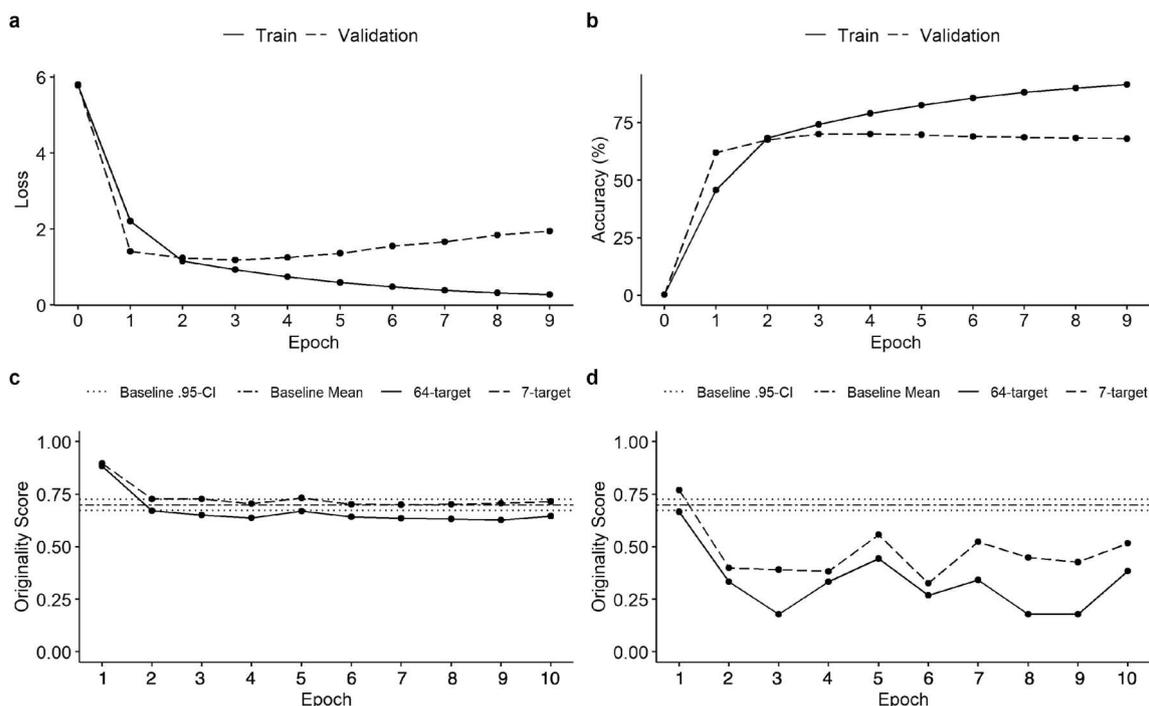


Fig. 7 **a** Loss curve of train and validation; **b** accuracy curve of train and validation; **c** mean originality curve for 64-target and 7-target sets; **d** minimum originality score curve for 64-target and 7-target sets

It shows that the originality scores of human composed excerpts have higher kurtosis, which implies that the model is less likely to generate excerpts within an narrow range of originality. The mean values denoted by the dashed lines also indicate the overall originality of generated excerpts is lower than the human composed excerpts.

For both algorithms, we see in Fig. 6a, b that the originality at the 2-bar level is low relative to the mean and 95%-confidence interval for the baseline, but this is to be expected, because the baseline was calculated at the 4-bar level. What we expect to see is the solid line—indicating algorithm originality at the 4-bar level—lie entirely inside that confidence interval. This is the case for MAIA Markov [9], but Music Transformer’s [19] mean originality level is entirely below this confidence interval, indicating it has issues with borrowing too heavily from the input on which it is trained.

Three typical worst case examples of copying are shown in Fig. 6c–e, with generated outputs on the left and original excerpts on the right. Figure 6c shows one from MAIA Markov having 42.9% originality associated with Beethoven’s String quartet no.6 in B-flat major, op.18, mvt.1, bars 61–64. And then, Fig. 6d shows one generated by the “best” checkpoint (checkpoint 3 with the minimum validation loss) of Music Transformer having 48.8% originality associated with Mozart’s String quartet no.13 in D minor, κ.173, mvt.1, bars 125–128. We found most of the

generated outputs in this stage with less than 50% originality are due to repeating the same note, which is also frequently found in Classical string quartets, and the model tends to start by reproducing this simple kind of pattern. Finally, Fig. 6e shows one generated by checkpoint 15 of Music Transformer having 9.9% originality associated with Beethoven’s String quartet no. 1 in F major, op.18, mvt.1, bars 233–234. In general, the model appears to be over-fit at this checkpoint. We infer from these originality reports and basic musicological interpretations that the results generated by Music Transformer gradually morph during training from reproduction of simple patterns (e.g., repeated notes) to verbatim use of more distinctive note sequences.

Incorporating Originality Reports into an Algorithmic Process: Originality Decreases as Epoch Increases

Here, we demonstrate the use of an originality report in the modelling process itself, as a means of analysing changes in originality as a function of model training or validation epoch. Music Transformer was used as an example of a deep learning model, with the train/validation set as in section “Determining the Baseline Level of Originality Within a Corpus”. To monitor the originality change along with the training process, 10 checkpoints including the initial point

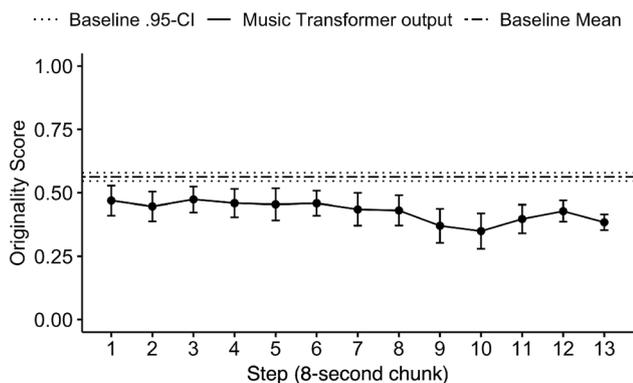


Fig. 8 Originality report based on geometric hashing for the Music Transformer algorithm, which shows the change in originality scores over the course of the excerpts obtained for Music Transformer at 8-s levels compared to the baseline mean and 95%-confidence interval

were saved. Again, we used each of them to generate 30 excerpts, to which the aforementioned originality report was applied. Afterwards, we calculated the mean value of those 30 originality scores for each checkpoint.

Figure 7a, b shows the change of loss and accuracy, respectively, over training. As mentioned previously, the standard training process would stop at epoch (checkpoint) 3, where the validation loss reaches a minimum, but we extended the training process further to more fully investigate the effect of training on originality. Figure 7c, d contains a dashed line indicating the baseline originality level of 0.699 for the string quartet data set. It is worth noting that Fig. 7c, d starts with epoch 1, instead of epoch 0, where the weights of networks were randomly initialised. Such difference is due to the near impossibility of forming notes from randomly generated sequences, because the serialisation method used by Music Transformer requires a matched pair of events to construct a note [19]. Thus, applying originality report for epoch 0 is unconsidered here. In Fig. 7c, mean originality score decreases as a function of model training epoch, but remains largely in the 95%-confidence interval of the baseline originality level of the corpus. Figure 7d shows more concerning, indicating that minimum originality score decreases to well below the 95%-confidence interval of the baseline originality level of the corpus. Originality decreases until epoch 3, and then it stays relatively flat afterwards. However, as with the discussion of Fig. 6c, d in the previous subsection, we found that the model’s borrowing still becomes more verbatim (or distinctive) after epoch 3, thus originality in a more general sense is still decreasing, a fact that is not immediately evident from Fig. 6, because the cardinality score does not consider distinctiveness, discussed further below.

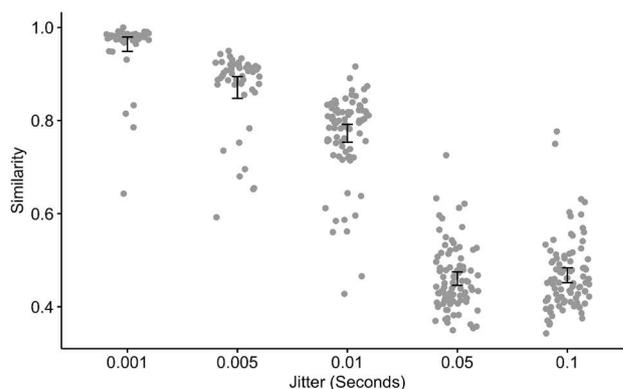


Fig. 9 Similarity changing with artificially added start time jitter. The similarity level generally decreases as jitter increases

Geometric Hashing: Producing Another Originality Report with a Different Similarity Measure, and Exploring Associated Parameters

In this section, we demonstrate the originality report with a similarity measure based on geometric hashing, and its adaptability to music with expressive timing.

This time, we use the trained version of Music Transformer from the original publication [19] (which used the MAESTRO train set), and generate 30 passages of “new” music. We then traverse each of these passages and collect 8-s excerpts with 4-s overlap. To adapt with expressive timing, the time difference calculated in this case is in seconds instead of crotchet beats. As shown in Fig. 8, Music Transformer’s [19] mean originality level is again entirely below the self-originality confidence interval of the corpus, again indicating it has issues with borrowing too heavily from the input on which it is trained.

The adaptability of our geometric hashing approach to data with expressive timing is already implicit in the results shown in Fig. 8, but we need to verify that: (1) when we define a query based on an excerpt of music that is drawn from (known to be in) the train set itself, we get a near-1 similarity score when the timing data in the query is perturbed by or subject to minimal jitter; (2) there is some reduction in similarity score as a function of jitter being increased from barely noticeable (around 1 msec) to barely recognisable as the same query (around 100 msec).

For each (time, pitch)-pair in a randomly selected query, we add random, uniform jitter with range $\pm x$ sec to the start time. In this case, we sample queries from the train set itself, and match to the lookup table calculated previously without jitter. We also verified that for no jitter and a bin size of 0.1 s (fixed for this analysis), each query is able to

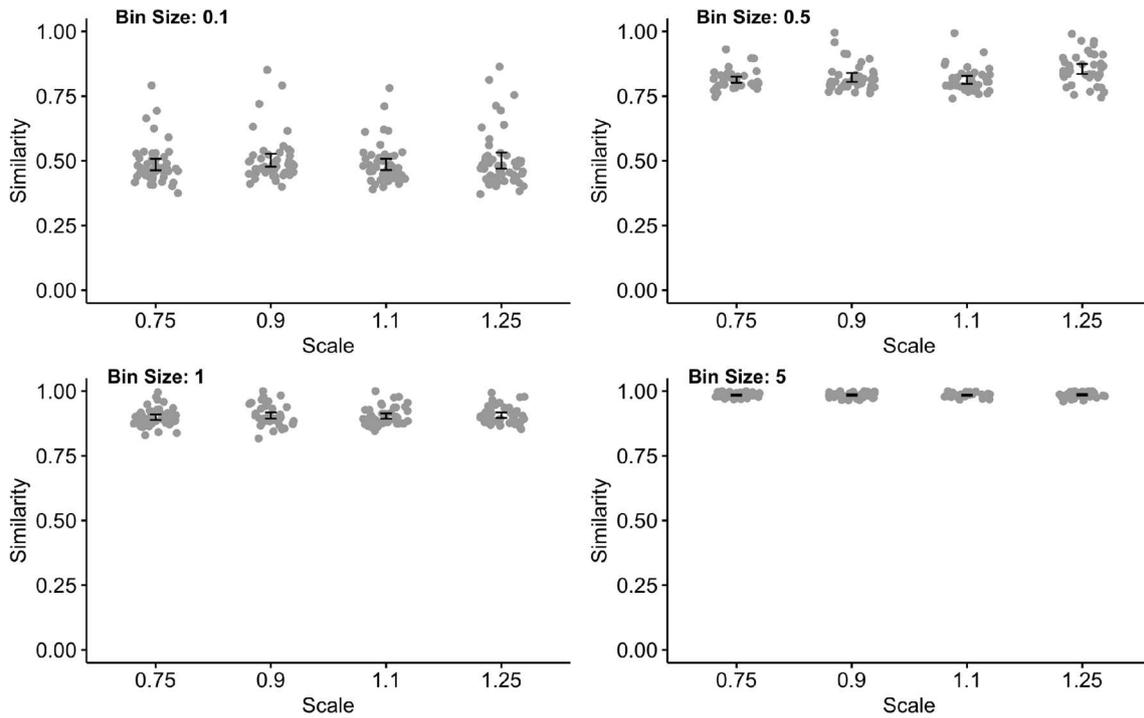


Fig. 10 Similarity changing with a set of bin sizes [0.1, 0.5, 1, 5], and for each with a set of scale factors [0.75, 0.9, 1.1, 1.25]

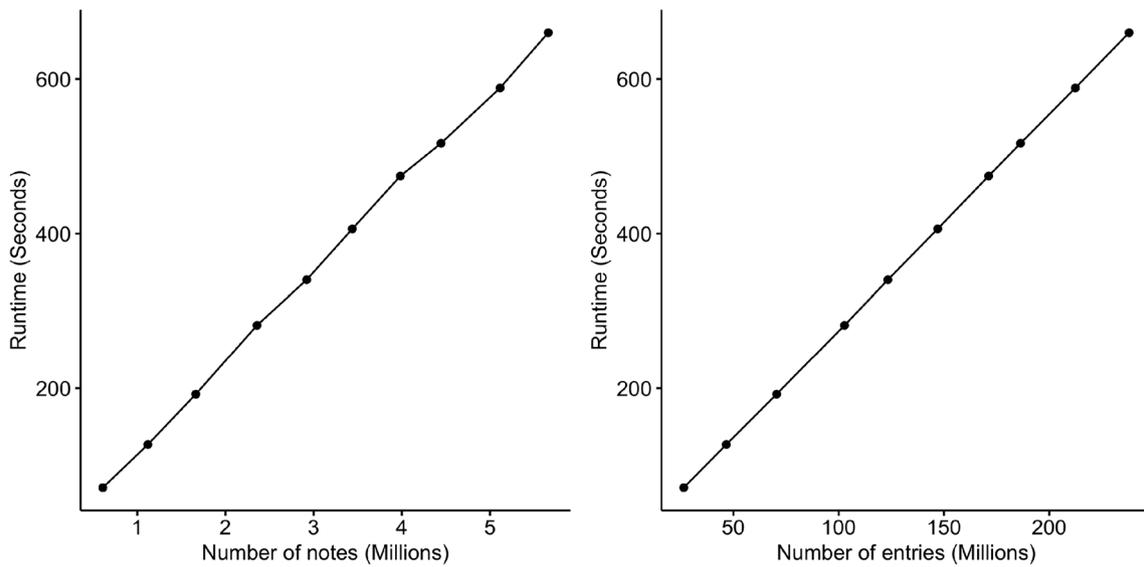


Fig. 11 Two line plots of runtime in seconds against number in millions of notes and entries created during lookup table building

produce maximal similarity 1 to the same piece from which is sampled.

As shown in Fig. 9, the similarity values decrease as the jitter increases, which is what we expected. It is worth noting that there is a slight increase in similarity score moving from 0.05 sec jitter to 0.1 s. This is because when we obtain

the top m similarity scores and their associated pieces, we use $m = 100$ and look for the similarity score associated with the piece from which the query was drawn. However, as jitter increases to 0.05 sec and beyond, we noticed that often this piece is not among the top 100 matches. In such cases, we report the maximal similarity that was found (to some

other piece), and hence the right-hand two entries in this plot might be slightly inflated, compared to one another and/or to the left-hand three entries.

To investigate the effect of bin size and scale factor on similarity, we run the same self-querying process with pieces in the train set. Here, we dependently test with bin sizes [0.1, 0.5, 1, 5] (seconds) and scale factors [0.75, 0.9, 1.1, 1.25]. The musical effect of applying a scale factor is that the query is slowed down (scale factor < 1) or sped up (scale factor > 1). The scale factor is applied to the start time of all notes in a query, for every bin size above. Maximal similarity 1 can be obtained without applying scale factor (i.e., multiplying by scale factor 1). As Fig. 10 shows, a difference in scale factor has little effect on similarity, whereas increasing bin size increases the similarity.¹¹ The latter result is an expected phenomenon, because with larger bin size, the approach is more likely to count as a false-positive a hash entry that is located further away from the precise start time.

Finally, we conduct a runtime test on building the lookup table, with ten data sets of increasing sizes. There are a total of 962 pieces in MAESTRO train set, the size of the subset ranges from 1 to 10 in multiples of 96.2 (ceiling function is applied). This test is run on Intel(R) Core(TM) i9-9920X CPU @ 3.50GHz and the runtime is recorded for plotting Fig. 11, which contains two line plots for runtime against number of notes and number of hash entries created. The plot on the left shows that the runtime increases linearly as the number of notes increases. Although, according to the Algorithm 1, the time complexity for searching triples is $O(n^3)$, the actual runtime is linear to the size of input. We deduce the difference is because the runtime for each piece is roughly the same, so the relation becomes linear in this practical case. The plot on the right shows the runtime increases as the number of created entries increases, and combined with the plot on left, it indicates the space complexity is also linear.

Discussion

This paper puts forward the notion that AI for music generation should result in outputs that imitate instead of merely copying original pieces, and highlights that checks of whether this is the case—what we refer to as the originality report—are often omitted. We introduce the methodology of the originality report for baselining and evaluating the extent to which a generative model copies from training

data. This originality report is parameterised by a particular similarity measure. Here we illustrate two similarity measures: a relatively straightforward note-counting approach based on the cardinality score [7, 39], and one based on fingerprinting score [1, 6, 42]. By designing other similarity measures, it becomes possible to adapt the methodology to have emphases on different musical dimensions or aspects of those dimensions. We first use a relatively small data set with rigid ontime information, and analyse outputs from two example models, one deep learning algorithm called Music Transformer [19] and one non-deep learning model called MAIA Markov [9]. For Music Transformer, we additionally analyse its output with the MAESTRO data set [18], which contains a larger amount of pieces and all with expressive timing information. The analyses illustrate the use of this methodology and the existence of music plagiarism in recent research.

We recognise Google Magenta for making their source code (e.g., for Music Transformer) publicly available, because it enables a level of scrutiny that has not always been possible for previous work in this field [31]. That said, the results indicate a phenomenon, wherein this type of deep learning language model gradually copies increasingly distinctive chunks from pieces in the train set, calling into question whether it really *learns* to generate. Deep learning often requires large data sets with a wide variety to increase the model generalisation ability [2]. However, this does not necessarily increase the likelihood of generating outputs with a wide variety. Particularly here, these results highlight that the Music Transformer model's [19] is not sufficiently original compared to the self-originality of the corpus on which it is trained. Stepping back somewhat from these specific results, we see several findings in this paper (e.g., those associated with Figs. 6b, d, e, 7, and 8) as evidence that current deep learning models for music generation might be good “data memorisers and regurgitators”, but fall short of being “original” or “creative”—not just qualitatively but as measured quantitatively by our originality report. More recent research found the information in training data can be retrieved from large language models, which highlights various issues of memorization [4]. Furthermore, using the conventional stopping criteria for the training process, the “best” model not only has a low level of originality, but also the quality of generated excerpts is low in the sense that the same note is repeated most of time (see Fig. 6d). Going forward, the field of deep learning needs to reconsider in what situations the conventional stopping criteria are appropriate: perhaps loss and accuracy should no longer be the only criteria when evaluating the model, because we need to prevent these models copying training data, especially when they are used increasingly in a “black-box” manner by practising musicians.

¹¹ Even if scale factor had a larger impact, it would be possible to iterate over several different likely scale factors during the matching process to take this into account, but the current analysis suggests this is not necessary, at least for the range of scale factors considered here.

Limitations

The size of the Classical string quartet data set used above is smaller than that used for the original work on Music Transformer [19]—MAESTRO [18]— and it is also quantised to a smaller set of time values. We address this issue using MAESTRO also, but another potential solution is to pretrain a generative model with a large data set to gain “general musical knowledge” and then finetune with a smaller style-specific data set [12, 14]. However, our first data set still represents a substantial amount of Classical music, and certainly enough to give a human music student an idea of the intended style, so if deep learning algorithms cannot operate on data sets of this size, then it should be considered a weakness of the deep learning approach rather than a limitation of our methodology.

The simplicity of the cardinality score has some appeal, but as noted in the review of existing work, it can mean that subtle variations along some musical dimension destroy any translational equivalence, giving a low cardinality score that is at odds with high perceived similarity. For instance, the expressive timing in the MAESTRO data set [18] constitutes such subtle variations along the dimension of start time, and make it ill-advised to use the cardinality score to assess the originality of an algorithm trained on these data. In addition, the cardinality score shares some general advantages of the geometric approach. However, in its current use, it is also not able to take into account the distinctiveness of excerpts being compared [6, 11]. For instance, Fig. 2 indicated an instance of similarity between Mozart and Haydn, but when we take into account how many Classical pieces end in this way, it is not a particularly distinctive or interesting example.

As mentioned, cardinality score does not work properly with subtle variations (e.g., expressive timing), while the symbolic fingerprinting using geometric hashing is able to detect similarities with such data. However, this complex approach also requires configuration of parameters (e.g., pitch/time difference range, bin size) to balance the efficiency and effectiveness. For instance, increasing the pitch/time difference range extends the coverage of distinctive fingerprints, but this also exponentially increases space and time complexity. In terms of distinctiveness, we observe some fingerprints have higher frequencies than others in a data set, which implies that highly frequent fingerprints correlate with the style of a music corpus, while infrequent ones make pieces distinctive from one another. However, the current version of similarity calculation, ratio of unique fingerprints, does not reflect these distributional characteristics.

Future Work

We would like to see the originality report method that we have developed be embedded into the training processes of

various music generation algorithms, to play a role as an advanced stopping criterion. Meanwhile, we will need to ensure that this criterion can still maintain the generalisability asserted by standard stopping criteria.

We will investigate the compatibility of the originality report method with model selection, which is often conducted as an outer loop of model training. Loss function engineering is a topic addressed in recent novel generating strategies (e.g., [16]), so it should be possible to merge high/low originality scores as rewards/penalties in training loss, to further investigate the problem that we have identified of language-based deep learning models appearing to be little more than powerful memorisers.

We will also explore further alternative similarity measures. Weighting of shift errors [7] and consideration of distinctiveness [11] could both address limitations mentioned above arising from the simplicity of the cardinality score. We will optimise the time performance of our geometric hashing approach [1], and investigate whether weighting the fingerprints based on their distribution in a data set can make the fingerprinting score reflect distinctiveness. This should mean originality reports can be generated for an algorithm trained on *any* music data, taking into account distinctiveness with respect to an underlying corpus.

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Zongyu Yin and Tom Collins. The first draft of the manuscript was written by Zongyu Yin, with the results section being developed by Tom Collins. All authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding The authors did not receive support from any organisation for the submitted work.

Availability of Data and Materials Data and materials used and collected from the study can be accessed via <https://osf.io/96emr/>.

Code Availability Code for running experiments and analyses can be accessed via <https://osf.io/96emr/>.

Declarations

Conflict of Interest The last author is a co-founder of the music collective Music Artificial Intelligence Algorithms, Inc. As such, he has an interest in the strong performance of the MAIA Markov algorithm. However, such interest does not affect using the introduced method to produce objective results. Apart from this, the authors have no relevant financial or non-financial interests to disclose.

Ethics Approval Not applicable.

Consent for Publication All authors consent to the publication of this work.

Consent to Participate Not applicable

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Arzt A, Böck S, Widmer G. Fast identification of piece and score position via symbolic fingerprinting. In: ISMIR; 2012. pp. 433–8.
- Barbedo JGA. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Comput Electron Agric.* 2018;153:46–53.
- Bengio Y. Practical recommendations for gradient-based training of deep architectures. In: *Neural networks: tricks of the trade.* Berlin: Springer; 2012. pp. 437–78.
- Carlini N, Tramer F, Wallace E, Jagielski M, Herbert-Voss A, Lee K, Roberts A, Brown T, Song D, Erlingsson U, et al. Extracting training data from large language models. 2020. [arXiv:2012.07805](https://arxiv.org/abs/2012.07805).
- Collins T. Improved methods for pattern discovery in music, with applications in automated stylistic composition. Ph.D. thesis, The Open University. 2011.
- Collins T, Arzt A, Frostel H, Widmer G. Using geometric symbolic fingerprinting to discover distinctive patterns in polyphonic music corpora. In: *Computational music analysis*, pp. 445–74. Springer. 2016.
- Collins T, Böck S, Krebs F, Widmer G. Bridging the audio-symbolic gap: the discovery of repeated note content directly from polyphonic music audio. In: 53rd international conference: semantic audio. Audio Engineering Society. 2014.
- Collins T, Coulon C. MAIA Util: an NPM package for bridging web audio with music-theoretic concepts. In: *Proceedings of the web audio conference*, pp. 47–52. Trondheim, Norway. 2019.
- Collins T, Laney R. Computer-generated stylistic compositions with long-term repetitive and phrasal structure. *J Creative Music Syst.* 2017;1:2. <https://doi.org/10.5920/JCMS.2017.02>.
- Collins T, Laney R, Willis A, Garthwaite PH. Developing and evaluating computational models of musical style. *AI EDAM.* 2016;30(1):16–43.
- Conklin D. Discovery of distinctive patterns in music. *Intell Data Anal.* 2010;14(5):547–54.
- Conklin D, Witten IH. Multiple viewpoint systems for music prediction. *J New Music Res.* 1995;24(1):51–73.
- Cope D. *Computer models of musical creativity.* Cambridge: MIT Press; 2005.
- Donahue C, Mao HH, Li YE, Cottrell GW, McAuley J. LakhNES: improving multi-instrumental music generation with cross-domain pre-training. In: ISMIR; 2019. pp. 685–92.
- Dong HW, Hsiao WY, Yang LC, Yang YH. Musegan: multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In: *Thirty-Second AAAI conference on artificial intelligence.* 2018.
- Elgammal A, Liu B, Elhoseiny M, Mazzone M. Can: creative adversarial networks generating “art” by learning about styles and deviating from style norms. In: 8th international conference on computational creativity, ICCO 2017. Georgia Institute of Technology. 2017.
- Graves A, Mohamed Ar, Hinton G. Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing, IEEE; 2013. pp. 6645–49.
- Hawthorne C, Stasyuk A, Roberts A, Simon I, Huang CZA, Dieleman S, Elsen E, Engel J, Eck D. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In: *international conference on learning representations.* 2019.
- Huang CZA, Vaswani A, Uszkoreit J, Simon I, Hawthorne C, Shazeer N, Dai AM, Hoffman MD, Dinculescu M, Eck D. Music transformer: generating music with long-term structure. In: ICLR. 2018.
- Janssen B, Collins T, Ren IY. Algorithmic ability to predict the musical future: Datasets and evaluation. In: ISMIR; 2019. pp. 208–15.
- Kingma DP, Ba J, Adam: A method for stochastic optimization. 2014. [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- Lewin D. *Generalized musical intervals and transformations.* Oxford: Oxford University Press; 2007. (originally published by Yale University Press, New Haven, 1987).
- Lowe DG. Distinctive image features from scale-invariant keypoints. *Int J Comput Vis.* 2004;60(2):91–110.
- Meredith D. The ps13 pitch spelling algorithm. *J New Music Res.* 2006;35(2):121–59.
- Meredith D, Lemström K, Wiggins GA. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *J New Music Res.* 2002;31(4):321–45.
- Müllensiefen D, Pendizich M. Court decisions on music plagiarism and the predictive value of similarity algorithms. *Musicae Sci.* 2009;13(1):257–95.
- Müller R, Kornblith S, Hinton GE. When does label smoothing help? In: *Advances in neural information processing systems;* 2019. pp. 4694–703.
- Neely A. Why the Katy Perry/Flame lawsuit makes no sense. 2020. <https://www.youtube.com/watch?v=0ytoUuO-qvg>. Accessed 30 Oct 2020.
- Oore S, Simon I, Dieleman S, Eck D, Simonyan K. This time with feeling: learning expressive musical performance. *Neural Comput Appl.* 2018;2018:1–13.
- Otzen E. Six seconds that shaped 1500 songs. 2020. <https://www.bbc.co.uk/news/magazine-32087287>. Accessed 30 Oct 2020.
- Pachet F, Papadopoulos A, Roy P. Sampling variations of sequences for structured music generation. In: ISMIR; 2017. pp. 167–73.
- Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015. [arXiv:1511.06434](https://arxiv.org/abs/1511.06434).
- Roberts A, Engel J, Raffel C, Hawthorne C, Eck D. A hierarchical latent vector model for learning long-term structure in music. In: *International conference on machine learning.* PMLR; 2018. pp. 4364–73.
- Ruiz A, Simon I. My only problem with Magenta's Transformer. *Magenta Discuss* Google Group. 2020. <https://groups.google.com/a/tensorflow.org/g/magenta-discuss/c/Oxiq-Gdaavk/m/uHIsQZKtBwAJ>. Accessed 30 Oct 2020.
- Selfridge-Field E. Conceptual and representational issues in melodic comparison. *Comput Musicol.* 1999;40:93.
- Stav I. Musical plagiarism: a true challenge for the copyright law. *DePaul J Art Tech Intell Prop L.* 2014;25:1.
- Sturm BL, Iglesias M, Ben-Tal O, Miron M, Gómez E. Artificial intelligence and music: open questions of copyright law and

- engineering praxis. In: Arts, vol. 8, p. 115. Multidisciplinary Digital Publishing Institute. 2019.
38. Todd PM. A connectionist approach to algorithmic composition. *Comput Music J.* 1989;13(4):27–43.
 39. Ukkonen E, Lemström K, Mäkinen V. Geometric algorithms for transposition invariant content-based music retrieval. In: *Proceedings of the international symposium on music information retrieval*; 2003. pp. 193–9.
 40. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. In: *Advances in neural information processing systems*; 2017. pp. 5998–6008.
 41. Volk A, Chew E, Margulis EH, Anagnostopoulou C. Music similarity: concepts, cognition and computation. *J New Music Res.* 2016;45(3):207–9.
 42. Wang ALC, Smith III JO. System and methods for recognizing sound and music signals in high noise and distortion (2012), patent US 8,190,435 B2. Continuation of provisional application from 2000. 2012.
 43. Wang X, Pang K, Zhou X, Zhou Y, Li L, Xue J. A visual model-based perceptual image hash for content authentication. *IEEE Trans Inf Forensics Secur.* 2015;10(7):1336–49.
 44. Widmer G. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artif Intell.* 2003;146(2):129–48.
 45. Yates P. *Twentieth century music: its evolution from the end of the harmonic era into the present era of sound.* New York: Allen & Unwin; 1968.
 46. Zauner C. Implementation and benchmarking of perceptual image hash functions. In: *Master's thesis, Upper Austria University of Applied Sciences.* 2010.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.