Neural Computing and Applications (2020) 32:969–979
https://doi.org/10.1007/s00521-018-3765-x

DEEP LEARNING FOR MUSIC AND AUDIO

Towards a *Deep Improviser:* a prototype deep learning post-tonal free music generator

Roger T. Dean[1,2] · Jamie Forth[3]

Received: 7 December 2017 / Accepted: 28 September 2018 / Published online: 15 October 2018

Abstract
Two modest-sized symbolic corpora of post-tonal and post-metrical keyboard music have been constructed: one algorithmic and the other improvised. Deep learning models of each have been trained. The purpose was to obtain models with sufficient generalisation capacity that in response to separate fresh input seed material, they can generate outputs that are statistically distinctive, neither random nor recreative of the learned corpora or the seed material. This objective has been achieved, as judged by *k*-sample Anderson–Darling and Cramer tests. Music has been generated using the approach, and preliminary informal judgements place it roughly on a par with an example of composed music in a related form. Future work will aim to enhance the model such that it deserves to be fully evaluated in relation to expression, meaning and utility in real-time performance.

Keywords Deep learning · Music · Post-tonal · Post-metrical · Improvisation

1 Introduction

Could a deep learning model of music function as a free improvising partner? Free improvisation is commonly mainly post-tonal (that is, most of the time it lacks emphasis on pitch structures that are hierarchical) [1, 2] and often post-metrical (that is, mostly it lacks hierarchical repetitive rhythmic structures) [3–5]. In other words, it is often very different from common practice Western music, or pop and rock (as illustrated in Online Resource 3 amongst Electronic Supplementary Material). An experienced free improviser in any artistic form is usually capable of responding to highly diverse, often unanticipated inputs in ways that are potentially equally diverse and sometimes unfamiliar even to the improviser [4, 6]. Experimental work has revealed something of the decision-making involved [7–9] (reviewed [10, 11]). By the same token, a free improviser may choose, for example, to adopt a tonal or metrical posture, or engage in blues-oriented phrases, at any time. Thus, free improvisation is not congruent with post-tonal and/or post-metrical music, but at the least includes them. Indeed, the implied conceptual ideal that a free improviser is always able to migrate away from their own conventions is of course unrealistic. In contrast to free improvisation, most algorithmic (computational) music generation systems have their own fixed heuristics, though sometimes responding to external events by using machine listening to transform the outputs of those otherwise unchanged heuristics [12–16]. It seems that in principle a deep learning neural net model based on appropriate corpora might learn a diverse enough set of statistical associations that it could be generatively seeded and sampled so as to function like a free improviser. This paper demonstrates a first step towards such a system.

Why do we focus on deep learning as a candidate generative framework, rather than, for example, variable order Markov chain or *n*-gram models [17, 18] or time series analysis models [19]? Essentially, it is the nonlinear activation components within neural nets, together with a possibly large number of input features, and the potential

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s00521-018-3765-x) contains supplementary material, which is available to authorized users.

✉ Roger T. Dean
roger.dean@westernsydney.edu.au

[1] MARCS Institute for Brain, Behaviour and Development, Western Sydney University, Sydney, Australia

[2] austraLYSIS, Sydney, Australia

[3] Department of Computing, Goldsmiths, University of London, London, UK

for the deep learning system itself (rather than the user) to determine which are the useful features for prediction that underlies the appeal. These features imply a capacity for very long-term hierarchical relationships to be grasped (which is more difficult to conceive with the mentioned alternative approaches). In addition, the developing evidence that layers within a deep net may take on distinctive tasks (for example, the identification of particular features of a visual image, for example, see the concise discussion in [20]) also suggests that deep learning may offer distinctive contributions to music generativity. This view is supported by an extensive review of the current approaches to music generativity, which nevertheless does not go deeply into comparisons with other approaches [21]; see also [22] for contemporary reviews of algorithmic music making.

Unlike the present paper, the previous deep learning music generation systems have mainly focused on generation of common practice instrumental music (using symbolic representations); see reviews [17, 21, 23, 24]. A more recent emphasis is on audio generation (using digitised wave form representations) [25, 26]. Considering the case of music with symbolic representation (and thus potentially conventional musical notation), the highly successful FolkRNN [27] produces music closely akin to Irish Folk music, with clear tonal and metrical features very much in common with it. Occasional audible perturbations to those features occur. Performance RNN is a recent output of the Google Magenta project [28]. It uses a quantisation of rhythm into 10 ms units, in which durations only up to 1 s are used, so as to create 'expressive timing' and dynamics. The online illustrative sound excerpts of Performance RNN (as in the training corpus of piano music used) are largely tonal, and the authors themselves describe the outputs somewhat dismissively as 'noodling'. One feature commonly lacking in deep learning generators is overt hierarchical structure, such as repetitions over a long time scale. In the work on audio generation from audio inputs, SampleRNN [25] introduces multiple levels of temporal hierarchy, but over less than a second. However, the principle can clearly be extended.

Given our intent to progressively assemble a *Deep Improviser,* a machine capable of free improvisation particularly in conjunction with human partners, we started with the target of multi-hand keyboard music and considered carefully the symbolic representation appropriate for our purpose. The resultant representation is detailed in the next section.

The paper proceeds as follows. In Sect. 2, we describe some of the long-term purposes of the work and how these lead to our adopted form of symbolic representation of keyboard music. We also summarise the key criteria we aimed to fulfil in our initial prototype system. Section 3 describes the generation of musical corpora for training the models, which in turn is described in Sect. 4. The generative step is discussed in Sect. 5, together with statistical data on the nature of the corpora, inputs and outputs. Section 6 provides discussion, conclusions and some pointers to future work.

2 Specific purposes and the musical representation

Keyboard music consists of two types of events: single notes and chords (which comprise several simultaneously sounded notes). Thus, we will refer to 'event' when we are making no distinction between these two types, and 'chord' or 'note' when we need to be specific. To accommodate our target, multi-hand post-tonal music, our intended model needs to allow chords to comprise any pitch combinations, and we decided that up to 10 pitches (notes) could be allowed in a chord, based on practical experience in keyboard improvisation. Some of the relevant music involves multiple algorithmic parts, or multiple keyboard players at one or two pianos, and hence, we refer to our target as 'multi-hand' keyboard music, noting that with four hands most (but not all) likely combinations of pitches, even if widespread across the keyboard, can be performed. To accommodate post-metrical features, our model also needs to permit continuous variation in the event duration (be it chord or note) and inter-onset interval (ioi) between events (time of event onset does not so clearly represent this feature, but rather requires differencing to generate it). We allowed these to be continuous variables, with an upper limit of 20 s (after which the sound of almost any piano note has died away). We chose further to represent pitch and key velocity (dictating acoustic intensity of the sounded note/chord) by continuous values bounded, respectively, by 0–120 (standard MIDI values, where the note middle C, normally termed C4, is 60) and 0–127, so that in the future these two could also be performed as continuous variables. This would allow continuously morphing microtuning and could avoid or embrace the concept of a dominant tuning system, such as that of most Western music. Note, for example, that post-tonal music does not always eschew tonality, just as 12-tone serialism, pioneered a century ago, often strives for atonality [29], but not always, and in any case does not always achieve it [1, 2]. Analogous flexibilities apply to timing and tuning issues. In sum, our system should be able to accommodate tonality or continuous pitches, and metricality or a time continuum, together with the conceivable intermediates.

So as illustrated in Fig. 1, each event is defined in the same way by a vector of 13 numbers: numbers 1–10 are pitches, so that a melody note just occupies location 1, and

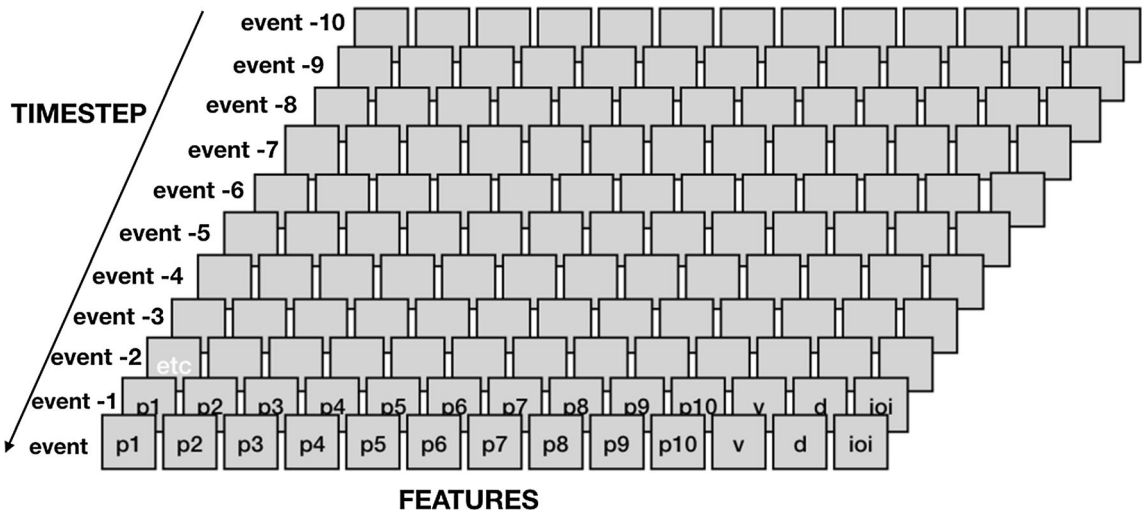

Fig. 1 Musical representation in the form of a single input matrix. Each event is represented as a vector of pitches (1–10), together with the corresponding single velocity (v), duration (d) and inter-onset interval (ioi) to the next event. When an input event is a 'note', only $p1$ has a positive number, and $p1$–10 are each − 1. When an event is a chord, several of $p1$–10 have positive number entries, the remainder 2–10 are occupied by a preset value of − 1, suitable for the following regression. An event comprising a chord of five notes would have five locations with pitch values and five locations still set at − 1. We organise the pitches of the notes to descend from $p1$ towards $p10$. We accept input pitch values from MIDI 0–120 as noted (the full range provided by some of the algorithms used in our Algorithmic Corpus). The model predictions can thus encompass both negative and positive values, and we assess the model precision in relation to these 'raw' predictions, but when realising musical outputs from predictions, we only allow pitch outputs from 12 to 113 (many pitch values below 12 or above 113 are audible, but essentially nondiscriminable from each other). Input vector location 11 is occupied by a single velocity value (MIDI range 0–127), but sonically realised outputs are constrained to 20–127, since values below 20 are again usually indiscriminable and mostly inaudible. Vector locations 12 and 13 in the representation each provide a continuous value (quantised to integer ms in the inputs), respectively, for event duration (note or chord), and similarly a value for inter-onset interval (ioi), the delay time until the next event. Note durations may be longer than ioi, so notes may overlap each other. Because the musical material being learned has strong time series autoregressive properties (see detail in Sect. 4), we included 10 lags of the input series 13-component vector together with the present event as the basis for outputting a prediction of the next 11 13-component vectors, including the next event. In other words, each input to the deep learning model has the shape 11 × 13 (as does each output).

Our representation currently involves simplifications, in that often notes of a given chord event have different durations that we do not include. Similarly, in this prototype we disregard keyboard pedalling (which is recorded in MIDI and can override the offset time of a sounded note). We plan to introduce both features into our system later.

We want to be able to manipulate/sample the model itself to produce varied outputs, eventually in real time. As a first step, we pursue the possibility of perturbing the outputs by seeding with new material (which in principle could be generated live while the model proceeds). In our preceding work on adventurous text generation [30], we also used sampling temperature (dictating the entropy of the sampling distribution from which a prediction is chosen) to enhance diversity of output and showed the success of this. In the present work on music generation, we have delayed this manipulation for future implementation.

3 Creating multi-hand keyboard corpora

We constructed two keyboard corpora for the training of our initial models, since we can find no prior symbolic corpus with the objectives and features we required (post-tonal and post-metrical). First, our 'Algorithmic Corpus' was made by concatenating 13 runs of six different compositional algorithms developed in the previous work by author RTD (an internationally active composer/improviser). Several of these algorithmic pieces were multi-strand in nature, that is, they have multiple simultaneous melodic strands as in chamber and orchestral music, as

well as chords vs notes. Most of the algorithms are interactive (so-called live algorithms). RTD also performed nine keyboard improvisations specifically to form the second, 'Improvised Corpus', using a Yamaha CP300 weighted touch sensitive keyboard at Queen Mary University of London (20170905). A quite separate single improvised piece recorded in 2016 on a Kurzweil weighted touch sensitive keyboard constitutes the 'Seed Piece', from which 11-event segments are later used to seed the generation of output from the formed models. This piece is not included in the Improvised Corpus. An audio excerpt of one piece from the Algorithmic Corpus and another from the Seed Piece (to provide an example of improvised keyboard playing) are provided within Electronic Supplementary Material, realised using the Pianoteq physical synthesis piano. Pianoteq is particularly suitable in the longer term because it can be used effectively with any tuning system, not only the conventional Western tunings we use here. Our previous work showed the utility of synthetic serial music corpora in understanding the information content of such music and in modelling its pitch features [2].

Table 1 shows the basic features of the constructed corpora and of the Seed Piece. PCA was merely used to provide an impression of the size of the major principal components, such that any major contrasts might be revealed. The table illustrates the distinctiveness of the three different materials in every respect bar the overlap of PCA component variances between the Algorithmic Corpus and the improvised Seed Piece. Further comparisons between the Improvised Corpus and the Improvised Seed Piece are illustrated later in Fig. 3.

4 Developing a deep learning model

As noted above, we aimed for a model which could be perturbed (that is, it is sensitive to external input seeds), so as to generate outputs distinct in statistical nature from its learned corpus and from the Seed Piece. So we took the avoidance of overfitting very seriously, to seek such flexibility (generalisability) in the generation phase. Especially given the limited size of our corpora, we considered that a model that at least beats so-called common sense (or naïve) predictions (see next paragraph for detail), but is not necessarily the most precise feasible, might have internalised statistical associations that could provide the basis for such flexible generation.

As is well known, most musical time series can be modelled well as an autoregressive process, implying that each event is substantially predicted by a series of immediately preceding events [31–33]. Indeed, the normal common sense prediction for time series which we adopted (often called the naïve model) is that the next event is similar to the last and so guessed as being the same [34]. We undertook some simple autoregressive time series analyses (TSA) on the corpora each taken as a single whole, to pre-establish that each of our feature series, of pitch (p), velocity (v), duration (d) and inter-onset interval (ioi), is highly autoregressive, with lags of around 10 previous events plus the present being significant predictors of the next. In comparison, there was little cross-predictive capacity, for example, from ioi or duration on to pitch (assessed both by univariate TSA, with ioi, etc. as predictors, and by vector autoregression, in which features may potentially be mutually influential: see [19] for discussion of such relationships in keyboard music). Consequently, our representation provided inputs to the deep learning

Table 1 Descriptive statistics of the two constructed corpora and the seed improvisation (to be used with the models for generative purposes)

Material	Number of events	Total number of notes	Mean notes per event	Ratio between the number of chords and the total number of events	PCA components 1 and 2, % variance explained, respectively	
Algorithmic Corpus	16,484	66,892	4.05	0.65	81.4	18.5
Improvised Corpus	13,466	34,397	2.56	0.56	70.5	28.7
Seed Piece (improvised)	214	1001	4.68	0.72	81.0	18.7

Note that very few chord pitch combinations are repeated at all, whereas the pitches of many notes are. Note also that because of the occurrence of some uniform columns of − 1 values (unoccupied pitch components), the PCA was performed on unscaled values, for illustrative purposes only. The Seed Piece was solely used to provide randomly selected 11-event sequences which triggered generative outputs from the deep learned models

model that were sequences of 11 events, each represented by the vector of 13 values described above. These inputs were used successively to predict the next event, and the model was trained by comparing prediction with actual. The first 2/3 of each corpus was used as the corresponding training set, to which the deep learning net was exposed. The remaining 1/3 was used as a 'validation' set, used to assess the precision of the learned model when exposed to previously unseen material (i.e. its generalisation capability). No learning was permitted each time the validation set was run (i.e. the weights in the net remained fixed). A stopping criterion was used during training, focused on optimising fit to the validation set, to minimise risk of overfitting the training set. Learning was stopped when 20 epochs had failed to improve the loss (the estimation of prediction precision) observed in the validation set.

Dilated convolutional neural nets (CNN) have recently emerged as powerful models of sequence structure (see for extended review and practical tutorial on deep learning using Keras, which we used as our coding platform: [20]). Given our modest recent success in using these for poetic text generation, we first considered stacked CNN alone for the Deep Improviser. Because of the apparently greater capacity of recurrent neural nets (RNN) and in particular those using the LSTM (long short-term memory) nodes, we then considered RNNs receiving outputs from an initial CNN layer. We optimised models on the Algorithmic Corpus and then solely tuned and fine-tuned (varying the learning rate) these for the Improvised Corpus, since these results were adequate for our purpose, and we were not determined as yet on utterly optimising all models. Figure 2 summarises the form of the CNN/RNN model.

We largely overcame overfitting by a combination of stringent dropout (parameter 0.5) at each layer, including both dropout and recurrent dropout in LSTM layers, together with L2 kernel regularisation at each layer (parameter 0.01), and by application of the stopping criterion defined above. Overfitting was judged by continuous monitoring of loss (mean squared error) in the training set and also in a withheld (unlearnt) validation set. These both reached their minima at a similar point, and hence, the use of a second unlearnt test set was unnecessary (because the danger of progressive adaptation of the model not only to the training, but also the validation set was minimised). This also allowed us to train on a larger portion of the corpora than would have been feasible if a second unlearnt test set had been used. Furthermore, given the limited size of the corpora, our concern was more for the ability to generate, than for maximal learning. There was no shuffling, because these are autocorrelated time series (shuffling on the basis of pre-estimated phrase structure may be of future interest). We used robust scaling [to address the asymmetric distributions of temporal values (shown later in Fig. 3)] and the very different ranges of the individual parameters, and then, we undertook some hyperparameter tuning (notably to trim down to the minimal size nets effective for our corpora), together with limited parameter fine-tuning using sequenced learning rates. Modelling was carried out with Keras and Theano (0.9). Bidirectional RNN was ineffective (as might be expected, bar the occurrence of significant retrogradation, where, for example, a pitch sequence may occur both forwards and backwards). Pitch augmentation (by transposing the materials to every possible relative position within an octave range) has been found effective in improving loss measures during modelling with tonal music [35], but was not so here.

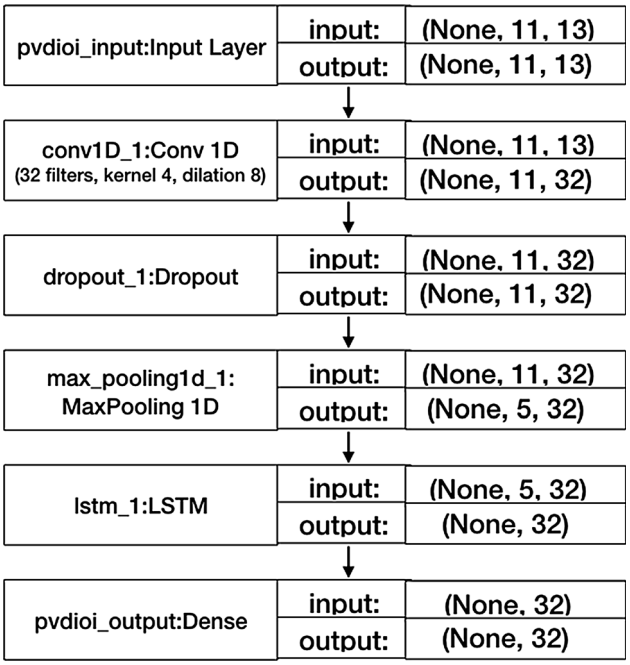

Fig. 2 CNN/RNN model. Based on the Keras format, the figure shows the sequential layers and summarises their nodes for the deep learning network adopted. 'Input' and 'output' describe the shape of the matrices handled at each sequential point, where 'None' indicates the total number of input/output matrices, which corresponds to the total number of rows in the corpus data being modelled. Batches of 128 inputs were used. Note that the 11 × 13 input and output vectors themselves are not represented in the diagram (see Fig. 1)

Table 2 summarises the performance of the resultant pair of models (one CNN alone and one CNN followed by RNN) as applied to both the Algorithmic and Improvised Corpora (different weights in each case, but identical model form). The data show consistently favourable comparisons with estimates of common sense predictions and reveal the capacity of the models to deal reasonably with previously unseen data (generalisability). Compared to the CNN-only model, the CNN–RNN gives an improved overall RMSE (root mean squared error) in the case of the Algorithmic Corpus, but not with the Improvised Corpus

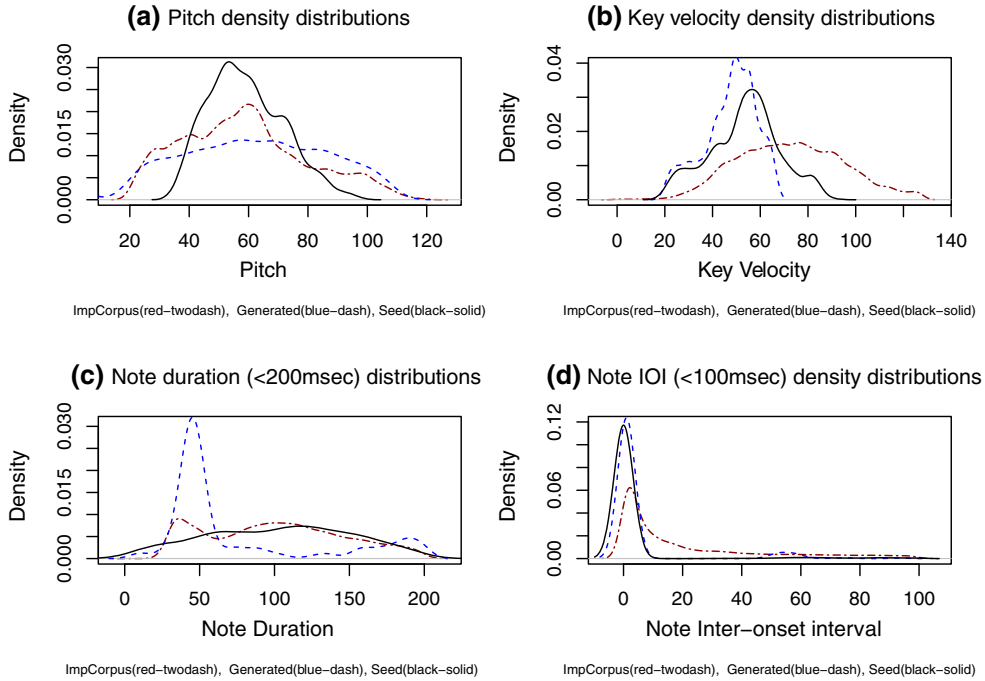

Fig. 3 Density distributions of pitch (**a**), key velocity (**b**), note duration (**c**) and note IOI (inter-onset intervals: **d**) for the Improvised Corpus, the Seed Piece and the generated output (1000 events). The descriptive characteristics of the corpus and the seed are summarised in Table 1. The distinctiveness of the generated output material is revealed as partly due to its broad pitch distribution (**a**) and partly to its emphasis on lower key velocities than either the seed or corpus (**b**). It has a dominant mass of durations around 50 ms (**c**), in part because of the simplification introduced in the model representation whereby the notes of a chord all have the same duration. This effect is supported by the relative preponderance in the generated output of chords over melody notes, resulting in a large mass around 0 ms note inter-onset interval, shared with the seed, but not the Corpus (**d**). Note that the graphed note duration distributions are truncated at 200 ms, and the inter-onset interval distributions at 100 ms to make the distinctions as clear as possible (no distinctions are visible in the much longer time values, and, for example, the IOI distributions range up to almost 20,000 ms)

Table 2 Performance of the selected deep learning models on the withheld validation set

Corpus/model	RMSE			Validation loss	Number of parameters
	Overall (vector of 13 values)	IOI	*p*1		
Algorithmic Corpus					
Naïve model	562.22	1004.23	24.55	N.A.	N.A.
CNN-only	397.04	415.66	15.49	95.25	7565
CNN/RNN	180.36	488.57	11.54	96.44	10,445
Improvised Corpus					
Naïve model	254.18	477.91	19.76	N.A.	N.A.
CNN-only	185.56	417.72	16.01	83.32	7565
CNN/RNN	196.30	450.24	19.20	80.90	10,445

The naïve model, as indicated in the text, was one in which the next event is predicted to be the same as the present one. The deep learning models were selected and optimised for hyperparameters, tuning and fine-tuning on the Algorithmic Corpus, and then adapted by tuning and fine-tuning on the Improvised Corpus. CNN, 64 filters, kernel of 4, dilation 8. CNN/RNN: CNN 32 filters, kernel of 4, dilation 8, RNN LSTM 32. N.A., not applicable. As indicated above, the loss measure is based on the robust scaling (allowing for the very different numerical scales of time and pitch) and not on the same scale as the larger RMSE values, which latter do correspond to the measurement units of the respective inputs

(further hyperparameter optimisation for this case could be attempted). The data also show that the temporal features contribute a large part of the RMSE, given their large values, while the RMSE attached to pitch 1 is respectably small. Given pitch ranges of 0–120 in the input, it can be seen that the RMSE of modelled pitch 1 in the several models (RMSE range 11–19 expressed in the pitch units) constitutes a maximum error of about 16%. Especially in

post-tonal music (where contour, that is whether the pitch sequence goes up or down at a given point, is likely far more important than precise pitch number), this seems to us quite usable. We noted above that the input representation ordered the pitches of a chord event from high ($p1$) to low (descending towards $p10$). We found that the predictions were similarly well ordered, and the number of predictions > 0 was closely parallel to the number of pitches in the input chord (where the remaining values in the input event vector $p1$–$p10$ set were -1 as described above). Correspondingly, the predictions included a similar number of values < 0. A consequence of this was that the RMSE for $p1$ (shown in Table 2) was generally 0–2 units larger than that for $p1$–10 taken together (data not shown), confirming that possible weighting within the pitch positions for the loss function would at most have the modest effect. Weighting the pitch and temporal components differently in the loss function for training (e.g. with respect to their relative variances) might be useful in the future. We could reasonably have improved the apparent RMSE by setting all $p2$–10 predictions that were < 0 to -1, but we chose not to do this.

Note that the purpose of adding the RNN with LSTM nodes (with only a c.50% increase in model parameters) is not solely to enhance the model precision, but also in the hope of enhancing model 'memory' (autoregressive and cross-parameter temporal relationships), such that it might predict longer sequences. We return to this issue in the next section.

5 Triggering, characterising and sounding distinctive outputs

Just as our models were trained on inputs of 11 events (each represented by a vector of 13 values) to predict the next event (in the same vector form), we planned to trigger outputs by using varied sequences of such vectors from an external (previously unseen) single improvised keyboard performance (214 events, comprising 1001 notes). We used the simplest form of prediction, in which the highest probability next event is selected (and making this more subtle will likely enhance the diversity of output we can obtain). So during generation each most likely prediction was added to the end of the seed, and the first seed member removed, so that after 11 predictions the new sequence constituting the next seed is composed entirely of model predictions. Thus, one important question is how frequently should one re-seed with a sequence from the external seed, which in turn brings up the question whether the models have sufficient memory to continue generating fresh sequences ad infinitum or whether they gradually regress to a fixed value set after a certain number of predictions: that is, they gradually converge on a prediction which remains thereafter constant. We expected this regression might occur in both models, particularly the CNN-only, both because of the limited capacity of CNN for temporally ordered sequences in comparison with LSTM (or gated response units, which we found to behave similarly in our system) units, and also the relatively small size of the learned corpora. The results of assessing this question were that both CNN and CNN/RNN models defined above regressed to a static value within about 60 predictions, after seeding once. This required that both the CNN and CNN/RNN models are reseeded very regularly for generative applications. While certain model modifications such as the introduction of residual connections (re-injection of earlier weights) may delay this regression to the mean, it is probable that larger corpora would help overcome this (and in turn, they may require larger deep learning nets). For assessments below, both CNN and CNN/RNN models were reseeded every 10 events with a randomly chosen (thus normally new) subsequence from the seed (reseeding every 11, being the number of input vector sequences in the learning phase, or 20 events was also functional). Varying the 'temperature' (as mentioned above) at which the learned output distribution is sampled may reduce the required frequency of reseeding. In total, 1000 events were generated: note again that each event may be a single note or a chord.

All outputs were floating point numbers which were approximated to the nearest integer. As mentioned above, predicted pitch and velocity values outside the ranges MIDI numbers 12–113 and 20–127, respectively, and predicted duration or ioi values less than 0 ms or more than 15,000 ms were rejected for the purposes of musical realisation, and the corresponding note was removed. Rejection was not used in the statistical analysis of the prediction performance and so did not distort the distributions of the respective parameters. Based on detailed prior empirical data on improvised keyboard performance [7, 36], all notes that occurred within 35 ms of an initial note were grouped together as a chord for realisation.

As we described previously [30], analysis of word outputs during text generation can be performed using word embeddings (vectorial representations of statistical word relations: reviewed [20]) or stylometry, based on relative word or n-gram frequencies, and we used the R 'stylo' package to successfully distinguish word output distributions in our generative poetry project. While chord2vec [35] and related modifications of word2vec [37] are useful in adopting a similar approach to tonal and metrical music, they are not applicable here because specific chord voicings rarely recur (though individual notes of course do), and the whole vector of $p1$–10, v, d, ioi essentially never recurs exactly, partly because of the continuous parameters

(finely quantised to 1 ms) involved in d and ioi. Thus, an alternative approach to assessing whether outputs are statistically distinctive or simply recreative has been adopted, in which we undertake univariate and multivariate testing of the question: what is the probability that the distribution of pitch (or velocity, duration, ioi) values observed in one case (e.g. the Algorithmic Corpus) and that observed in another (e.g. the generated output from the Algorithmic Corpus when seeded with the external improvised sequences) both arise from a parent distribution (which remains unspecified in nature). We undertook the statistical analyses in the R. For this purpose (and for playback), we reorganised the pitches of each chord into MIDI sequences of individual notes (for this assessment we are not considering time series sequential relationships). The Anderson–Darling test can determine this value, and Table 3 shows as an example the relevant results from the simpler CNN-only model based on the Algorithmic and on the Improvised Corpora, confirming that seeding with the external sequences from an improvised piece (a piece which is not included within the Improvised Corpus) is effective in driving generation which is not simply distributed the same way as either the learned corpus or the seed. The multivariate Cramer test, assessing whether two distributions are distinct, supports this conclusion where undertaken, considering all the parameters $p1$–10, v, d, ioi simultaneously in their chord/note event representation. Because the R Cramer algorithm generates very large numbers (which can outstrip the range permissible with the 32 bit number representations that R normally expects), it was necessary to undertake the test with subsets of the data. For example, it gave $p = 0$ for comparing 3000 sequential notes from the Improvised Corpus and from the deep learned CNN model of that corpus, seeded by the external improvised piece.

Figure 3 illustrates some of the distributions that are included in Table 3, specifically those from the seeded Improvised Corpus modelled by the CNN-only, and shows that its implications are visually plausible. Separate Anderson–Darling analyses (not shown) of the outputs from the CNN/RNN model of the Algorithmic Corpus (and the parallel analyses with the CNN–RNN Improvised Corpus model) support a general conclusion: that our approach permits the generation of sequences statistically distinct from either the learned corpus or the input seed distribution which are yet organised rather than random.

Going from statistical distinction to substantive human evaluation of computational artistic generativity is a hugely difficult task [38–41] as for that matter is evaluation of (manually) composed work, and there is also an argument that computational creativity (to which this paper potentially contributes) should be assessed in relation to its own specified objectives, partly or even solely by an internal mechanism [42]. In a previous paper on generativity with time series models [19], we elaborated in some depth on possible approaches which could minimise the psychological 'demand' commonly imposed in human listening tests (for example, avoiding reference to computational vs. compositional origins and hence the biases these commonly elicit). In addition, we pointed to the complexity of

Table 3 Output distinctiveness: univariate testing for possible statistical common origins of corpora, seed, and generated outputs, based on note sequences

Generator model	Comparison distribution origin	Anderson–Darling statistic T.AD, and (probability of origin from a common distribution reported by R) for the specified distribution parameter			
		Pitch distribution	Key velocity distribution	Note duration	Note inter-onset interval
Algorithmic Corpus	Generated output	734.2 (< 0.001)	394.7 (< 0.001)	615.8 (< 0.001)	1370 (0)
	Input seed	1090 (0)	1123 (0)	94.72 (< 0.001)	138.3 (< 0.001)
Improvised Corpus	Generated output	135.4 (< 0.001)	2670 (0)	694.3 (< 0.001)	2210 (0)
	Input seed	909.1 (0)	666.4 (< 0.001)	146.6 (< 0.001)	203.2 (< 0.001)

The CNN-only models of the Algorithmic and the Improvised Corpus were seeded with sequences of the external improvised piece (not included in the Improvised Corpus). The properties of the corpora and the seed are summarised in Table 1. The generated outputs are 1000 events. The Anderson–Darling k-sample test was performed in R with package kSamples. Its statistic T.AD is (AD criterion − mean)/sd, and there are separate versions for discrete and continuous distributions (quoted accordingly above). The statistic then provides a probability that the two distributions considered could come from a shared parental distribution (whose nature is not determined). These univariate measures were based on notes (and not chords, for reasons discussed in the text). The corpora and the seed were also mutually distinct judged by this test. p, pitch (encompassing all of $p1$–$p10$ sounded pitches and disregarding the unsounded values), v attack velocity, d note duration, ioi inter-onset interval

evaluating an output which occurs simultaneously with a live improvisation or an enunciation of a pre-composed element. While we intend to grapple fully with this in due course, for the time being we allowed ourselves a preliminary informal test: 21 researchers listened to three unidentified items of post-tonal and post-metrical keyboard music in a group setting. The items were 75–100 s in length and presented as 'multi-hand keyboard music' (and so it was pointed out that it was not necessary to assess whether what was heard was feasible for a single human to play). Listeners were informed that one piece was composed manually, one generated by a deep learning model with seeding, and one was composed algorithmically. The first item was an extract of composer Morton Feldman's *Piano Four Hands* (1957–1958, from the Etcetera KTC2015 CD performed by Roger Woodward), the second an extract of a Deep Improviser product and the third taken from an algorithmic piece included in the corpus it had learned (the latter two extracts are provided as supplementary material, while the first cannot be published here for reasons of copyright, but is available). After all items had been heard, votes were called first for preferred item, being, respectively, 3:3:15 (i.e. a preference for the algorithmic piece) and then for which piece was the deep learned product 11:10:0 (i.e. an equivocation between Feldman and Deep Improviser). There were only a few people amongst this group who had prior exposure to music of this kind, but in a separate group of three friends, all familiar with such work, when given the same test and scenario, the scores were for preferred item, 1:1:1 (no strong preference), and identifying the Deep Improviser piece, 1:0:2 (this time, an equivocation between Feldman and algorithmic composition). Thus, the Deep Improviser was not readily identified, and the algorithmic piece was overall preferred amongst the three, but the Deep Improviser was competitive with the human composer. We view this at least as support for the utility of our approach (RTD is a great admirer and was an acquaintance of the composer).

6 Discussion, conclusions and future work

Overall, our prototype *Deep Improviser* shows the initial signs of success: it can generate outputs distinct from its learned corpus or input seeds that nevertheless have commonality with them. Clearly this distinctiveness can likely be dramatically enhanced by control of sampling temperature (and this control might be powerful if used in a context-dependent manner). There are numerous limitations to the model so far, of which perhaps the most obvious are the relatively small corpora (though this is an advantage for any music maker wanting to establish their own model and corpora) and the sparseness of occupancy of the $p1$–10 part of the vectorial representation of an event (together with the lack of distinct duration values for individual notes of a chord). On the other hand, one-hot encoding (where, for example, pitches 12–113 would be represented each by a vector of 101 zeros with a 1 at the spot in the vector corresponding to the pitch) is also a sparse representation, yet has many benefits including categorical prediction and may be useful here (it is used as the basis of Performance RNN encoding). We have raised the issue of contour above, and it is apparent that accuracy of contour could be used as part of a loss function, regardless of tuning system or quantisation. Similarly, quantising durations, for example, at the observed 35 ms cut off between a succession of notes and a chord, may also be valuable. The fact that MIDI represents chords as a succession of notes separated by 0 or a few ms in timing also indicates that considering that we found it necessary for some purposes above to interconvert recorded chords and notes, there may be a case for generating chords as sequences of (almost simultaneous) notes rather than as such and modelling accordingly. This would also invite a hierarchical conditional model in which the first prediction is whether an event is a note or a chord, and the subsequent predictions then evaluate the chosen case (note or chord expressed as a rapid sequence of notes). This may ensure a wider range in the relative occurrence of chord versus melody notes in the generated outputs: as currently the chord:note ratio is commonly quite high. It would also present a pathway towards production of multiple parallel streams of events.

The time series analyses assessing relations between p, v, d and ioi mentioned above also revealed limited mutual influences in comparison with autoregressive influences, with minor exceptions. This suggests that a multi-input multi-output (branched) deep learning model, with only certain influences permitted, may provide a more accurate model, and one in which the weighting of the loss determined on the different components of the prediction (p, v, d, ioi) might be varied according to their variability and relative importance, analogous to approaches developed with multidimensional Markov models of music such as IDyOM [43]. This is underpinned by the fact that the distributions of the temporal features are very different from those of pitch and velocity. We will consider transformations based on cumulative density functions, identification of repetitions and geometrical structures within perceptually grounded representations as possible means of reducing the complexity of those data [44, 45], particularly for further analyses of outputs.

In the future development of this project, we want to create and use a system operative in real time, and given pre-learned models, this is already feasible. It will also be

possible to fit and update models in real time over an accumulating performed input, at the same time as generating from the current model with seeding and sampling. In our previous work, we have demonstrated the utility of analytical autoregressive multivariate time series models as generators themselves and constructed a system operative in real time [19]. The present *Deep Improviser*, while yet shallow, when comprised of CNN and RNN probably imitates some of the sequential aspects of time series models. But we also plan in the longer run to integrate *Deep Improviser* with algorithmic approaches based on information theoretic and perceptual decision-making models [46, 47].

Compliance with ethical standards

Conflict of interest The authors declare they have no conflict of interest.

References

1. Chew E (2005) Regards on two regards by Messiaen: post-tonal music segmentation using pitch context distances in the spiral array. J New Music Res 34(4):341–354
2. Dean RT, Pearce MT (2016) Algorithmically-generated corpora that use serial compositional principles can contribute to the modeling of sequential pitch structure in non-tonal music. Empir Musicol Rev 11(1):27–46
3. Jost E (1974) Free jazz, English edn. Universal, Graz
4. Bailey D (1992) Improvisation, its nature and practice in music, revised edition. British Library (first published 1980), London
5. Pressing J (2002) Free Jazz and the avant-garde. In: Cooke M, Horn D (eds) The Cambridge companion to jazz. Cambridge University Press, Cambridge, pp 202–216
6. Smith H, Dean RT (1997) Improvisation, hypermedia and the arts since 1945. Routledge, London
7. Dean RT, Bailes F, Drummond J (2014) Generative structures in improvisation: computational segmentation of keyboard performances. J New Music Res 43(2):224–236
8. Dean RT, Bailes F (2016) Relationships between generated musical structure, performers' physiological arousal and listener perceptions in solo piano improvisation. J New Music Res 45(4):361–374
9. Beaty RE (2015) The neuroscience of musical improvisation. Neurosci Biobehav Rev 51:108–117
10. Dean RT, Bailes F (2014) Cognitive processes in musical improvisation. In: Lewis GE, Piekut B (eds) Critical improvisation studies, vol 1. Oxford University Press, New York, pp 39–55. https://doi.org/10.1093/oxfordhb/9780195370935.9780195370013
11. Vuust P, Kringelbach ML (2017) Music improvisation: a challenge for empirical research. In: Ashley R, Timmers R (eds) Routledge companion to music cognition. Routledge, New York, pp 265–275
12. Rowe R (1993) Interactive music systems. Machine listening and composing. MIT Press, Cambridge
13. Lewis GE (2000) Too many notes: computers, complexity and culture in voyager. Leonardo Music J 10:33–39
14. Dean RT (2003) Hyperimprovisation: computer interactive sound improvisation; with CD-Rom. A-R Editions, Madison
15. Edwards M (2011) Algorithmic composition: computational thinking in music. Commun ACM 54(7):58–67
16. Herremans D, Chuan C-H, Chew E (2017) A functional taxonomy of music generation systems. ACM Comput Surv (CSUR) 50(5):61-33
17. Nierhaus G (2009) Algorithmic composition: paradigms of automated music generation. Springer, New York
18. Pachet F (2003) The continuator: musical interaction with style. J New Music Res 32(3):333–341
19. Dean RT (2017) Generative live music-making using autoregressive time series models: melodies and beats. J Creat Music Syst 1(2):1–19
20. Chollet F (2017) Deep learning with python. Manning Electronic Advanced Publication, Shelter Island
21. Briot J-P, Hadjeres G, Pachet F (2017) Deep learning techniques for music generation—a survey. arXiv:170901620
22. McLean A, Dean RT (2018) The Oxford handbook of algorithmic music. Oxford University Press, New York
23. Fernández JD, Vico F (2013) AI methods in algorithmic composition: a comprehensive survey. J Artif Intell Res 48:513–582
24. Colombo F, Muscinelli SP, Seeholzer A, Brea J, Gerstner W (2016) Algorithmic composition of melodies with deep recurrent neural networks. arXiv:160607251
25. Mehri S, Kumar K, Gulrajani I, Kumar R, Jain S, Sotelo J, Courville A, Bengio Y (2016) SampleRNN: an unconditional end-to-end neural audio generation model. arXiv:161207837
26. Engel J, Resnick C, Roberts A, Dieleman S, Eck D, Simonyan K, Norouzi M (2017) Neural Audio synthesis of musical notes with WaveNet autoencoders. arXiv:170401279
27. Sturm BL, Ben-Tal O (2017) Taking the models back to music practice: evaluating generative transcription models built using deep learning. J Creat Music Syst 2(1):27
28. Simon I, Oore S (2017) Performance RNN: generating music with expressive timing and dynamics. Magenta Blog. https://magenta.tensorflow.org/performance-rnn. Accessed 6 Dec 2017
29. Huron D (2006) Sweet anticipation. MIT Press, Cambridge
30. Dean RT, Smith H (2018) The character thinks ahead: creative writing with deep learning nets and its stylistic assessment. Leonardo 1–2 (**online, just accepted**)
31. Dean RT, Bailes F (2010) Time series analysis as a method to examine acoustical influences on real-time perception of music. Empir Musicol Rev 5:152–175
32. Serra J, Kantz H, Serra X, Andrzejak RG (2012) Predictability of music descriptor time series and its application to cover song detection. IEEE Trans Audio Speech Lang Process 20(2):514–525
33. Gingras B, Pearce MT, Goodchild M, Dean RT, Wiggins G, McAdams S (2016) Linking melodic expectation to expressive performance timing and perceived musical tension. J Exp Psychol Hum Percept Perform 42(4):594–609
34. Enders W (2004) Applied econometric time series, 2nd edn. Wiley, Hoboken
35. Madjiheurem S, Qu L, Walder C Chord2Vec: learning musical chord embeddings. In: Proceedings of the constructive machine learning workshop at 30th conference on neural information processing systems (NIPS'2016), Barcelona, Spain, 2016, paper 5, pp 1–5
36. Pressing J (1987) The micro-and macrostructural design of improvised music. Music Percept 5(2):133–172
37. Herremans D, Chuan C-H (2017) Modeling musical context using Word2vec. In: First international workshop on deep learning and music, pp 11–18

38. Pearce M, Wiggins G (2001) Towards a framework for the evaluation of machine compositions. In: Proceedings of the AISB'01 symposium on artificial intelligence and creativity in the arts and sciences, Citeseer, pp 22–32
39. Jordanous A (2012) A standardised procedure for evaluating creative systems: computational creativity evaluation based on what it is to be creative. Cogn Comput 4(3):246–279
40. Agres K, Forth J, Wiggins GA (2016) Evaluation of musical creativity and musical metacreation systems. Comput Entertain 14(3):1–35
41. Lamb C, Brown D, Clarke C (2017) Incorporating novelty, meaning, reaction and craft into computational poetry: a negative experimental result. In: Proceedings of 8th international conference on computational creativity, ICCC
42. Wiggins GA, Forth J (2017) Computational creativity and live algorithms. In: McLean A, Dean RT (eds) The Oxford handbook of algorithmic music. Oxford University Press, New York (**in press**, 2018)
43. Pearce MT, Wiggins GA (2006) Expectation in melody: the influence of context and learning. Music Percept 23:377–405
44. Forth J, Wiggins GA (2009) An approach for identifying salient repetition in multidimensional representations of polyphonic music. In: Chan J, Daykin JW, Sohel Rahman M (eds) London algorithmics 2008. College Publications, London, pp 44–58
45. Forth J, Wiggins GA, McLean A (2010) Unifying conceptual spaces: concept formation in musical creative systems. Mind Mach 20(4):503–532
46. Wiggins GA, Forth J (2015) IDyOT: a computational theory of creativity as everyday reasoning from learned information. In: Besold TR, Schorlemmer M, Smaill A (eds) Computational creativity research: towards creative machines. Springer, New York, pp 127–148
47. Forth J, Agres K, Purver M, Wiggins GA (2016) Entraining IDyOT: timing in the information dynamics of thinking. Front Psychol 7(1575):1–19