Deep Artificial Composer: A Creative Neural Network Model for Automated Melody Generation

Florian Colombo(✉), Alexander Seeholzer, and Wulfram Gerstner

School of Computer and Communication Sciences and School of Life Sciences, Brain-Mind Institute, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
florian.colombo@epfl.ch
http://lcn.epfl.ch

Abstract. The inherent complexity and structure on long timescales make the automated composition of music a challenging problem. Here we present the Deep Artificial Composer (DAC), a recurrent neural network model of note transitions for the automated composition of melodies. Our model can be trained to produce melodies with compositional structures extracted from large datasets of diverse styles of music, which we exemplify here on a corpus of Irish folk and Klezmer melodies. We assess the creativity of DAC-generated melodies by a new measure, the novelty of musical sequences, showing that melodies imagined by the DAC are as novel as melodies produced by human composers. We further use the novelty measure to show that the DAC creates melodies musically consistent with either of the musical styles it was trained on. This makes the DAC a promising candidate for the automated composition of convincing musical pieces of any provided style.

Keywords: Automated music composition · Deep neural networks · Sequence learning · Evaluation of generative models

1 Introduction

If music composition *simply* consists of applying rules, where does this leave musical creativity? Modern composers trying to escape the rules and structural constraints of traditional western music had to confront themselves with their inevitable presence [1] – by trying to flee from some limitations, we fall into others. While considered a drawback by some composers, others saw the very well defined structure of music as a possibility for its formalization [2].

The generation of music according to its inherent structure goes back as far as Ada Lovelace, who saw in the analytical engine of Charles Babbage, the ancestor of computers, the potential to *"compose elaborate and scientific pieces of music"* [3]. In the present age of machine intelligence, many approaches have been undertaken to algorithmically write musical pieces [4]. A large challenge

J. Correia et al. (Eds.): EvoMUSART 2017, LNCS 10198, pp. 81–96, 2017.
DOI: 10.1007/978-3-319-55750-2_6

for music production algorithms is to be trainable, allowing them to learn the structural constraints of a given body of music, while at the same time producing original musical pieces adhering to these constraints without simply replaying learned motifs.

We present here the Deep Artificial Composer (DAC), a trainable model that generates monophonic melodies close to, but different from, tunes of a given musical style, which still carry well-defined musical structure over long timescales. To achieve this, the DAC employs multi-layer recurrent neural networks consisting of variants of Long Short-Term Memory (LSTM) units [5] to implement a generative model of note transitions. Networks based on LSTM units and their variants [6,7] have yielded impressive results on tasks involving temporal sequences, such as text generation [8,9] or translation [10,11]. Importantly, when used as generative models, neural networks have been able to produce artistic and nonartistic data close to real data or human art [12,13]. A prominent recent example was able to extract and apply arbitrary painting styles to any given picture [14].

Earlier attempts at extracting and generating the complex temporal structures of music with neural networks do not yet produce convincing musical pieces, mainly because of the lack of long timescale musical structure [15,16] and creativity/novelty of the produced pieces [17]. In particular, earlier work also based on LSTM networks proposed to improvise patterns of notes constrained on the pentatonic scale and harmony for blues improvisation [18], or to reproduce exactly a unique melody [17]. While both of these works are promising for automated melody generation, it remained unaddressed how these neural networks could generalize from arbitrary corpora to compose new songs in similar styles, carrying a shared structure with the melodies in the given training corpus.

The DAC presented here is an extension of the generative model of music presented in [19]. There, the authors introduced a neural network that could learn some of the temporal dependencies between a note in a melody and the history of notes in the same melody. Also, they introduced the separation of pitch and duration features of musical notes to explicitly model their relation with two different RNNs. Here, we introduce additional network layers, a more generalizable processing of melodies, as well as a different training protocol. Importantly, we demonstrate that we can train the model on a corpus containing two inherently different musical styles. By evaluating on a large scale the melodies generated by DAC networks with a measure of musical novelty, we show that the DAC can learn to compose consistent songs in different scales and styles.

We begin by introducing the model in Methods as part of a general formalization of iterative sequence generation, motivating and comparing the DAC with other generative models. In the Results section, we show the evolution of the DAC networks during training and present our measure of novelty that we then use to assess the melodies generated by the network. This allows us to show that the generated melodies are close to, but different from, melodies from the learned musical corpus, and classify the style of generated melodies. Finally, we present and analyze two example melodies that the DAC imagined.

2 Methods

Data Processing and Representation. Melodies in symbolic notation (`abc` notation [20]) are converted to `MIDI` sequences of notes, including the full note sequences resulting from symbolic repetitions with possibly different endings (e.g. "prima/seconda volta"). The Python package music21 [21] is then used to extract the corresponding sequences of notes, which are used to build integer alphabets of possible pitches and durations, into which melodies are translated (see Fig. 1**A**). In addition, silences are encoded as an additional alphabet entry in the pitch representation, while song endings are represented as additional entries in both the duration and pitch alphabets.

For a given corpus of melodies, we first construct the alphabets that contain all the pitches and durations values present in the entire dataset and map each of them to a unique integer value. We then represent the notes of each song with respect to these alphabets, thus producing two sequences of integers (pitches and durations) for each song. Songs containing very rare events (total occurrence in all songs < 10) in terms of either duration or pitch are removed from the dataset. The probability of very rare notes occurring in generated melodies (see below) will be closely related to the number of their occurrence in the training set – by removing songs containing these rare notes we reduce the alphabet sizes, which leads to a smaller number of parameters and a decreased time needed to train the model, without affecting the notes occurring in generated melodies much. Finally, we remove duplicate songs from the dataset (determined by unique pitch sequences).

The preprocessing of melodies presented here is more general than the one presented in an earlier model [19], where the `abc` notation was used directly. First, this makes the implementation of our model applicable to the more universal and more widely available `MIDI` format. Second, in contrast to the representation used there, we *do not transpose songs* into C Major/A minor, in order to retain and learn the actual key of songs. It is worth noting that this data normalization could be performed for smaller datasets to increase the number of examples of each pitch transition, at the cost of information about musical key.

To represent pitches and durations as input to DAC networks, we use a one-hot encoding scheme. This consists of mapping integer values i (for pitch or duration) to binary vectors where each entry is 0 except the one at dimension i, which is set to 1. This input vectors will later be denoted by $\mathbf{p}^s[n]$ (for pitch) and $\mathbf{d}^s[n]$ (for duration) of the n^{th} note in song s.

General Statistical Formalism for Melody Generation. We considering a monophonic melody as a sequence of N notes $x[1:N]$. The DAC is built under the assumption that each note $x[n]$ is drawn sequentially from a probability distribution (the "note transition distribution") $T[n]$ over all possible notes. Generally, for a melody s, the transition distribution of note at time step n will depend on song dependent inputs $\Phi^s[n]$ (e.g. the history of notes up to note n)

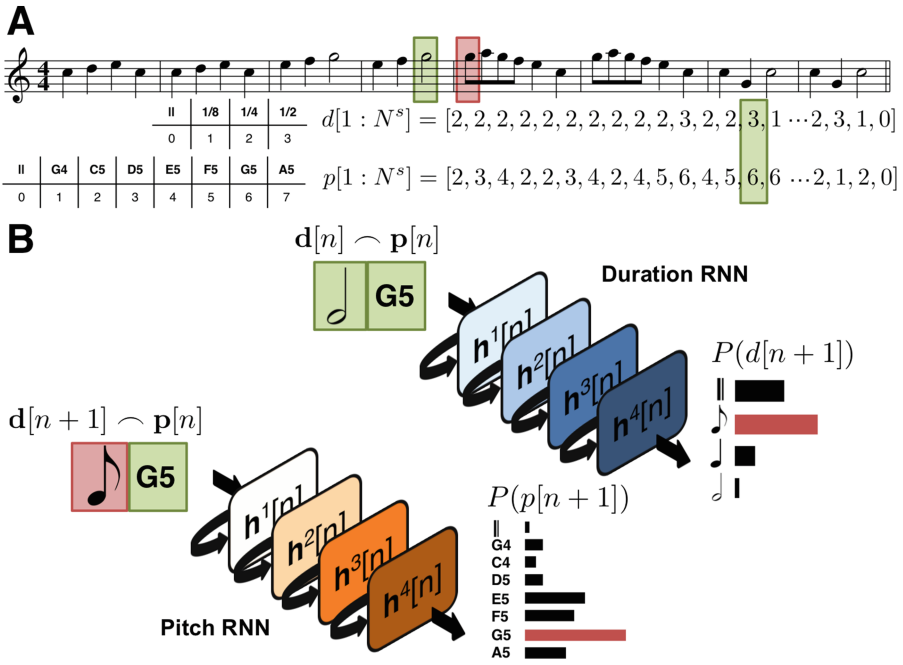

Fig. 1. Deep Artificial Composer networks: (**A**) Representation of the nursery rhyme *Brother John* in the DAC by two sequences of integers $d[1:N^s]$ and $p[1:N^s]$. Each integer in the top sequence corresponds to a single duration given by the duration alphabet to the left. Similarly, each integer in the bottom sequence corresponds to a single pitch given by the alphabet of pitches to the left. (**B**) The flow of information in DAC networks. The current duration $d[n]$ and pitch $p[n]$ are inputs to the duration network (bottom), while the pitch network (top) receives as inputs the current pitch $p[n]$ and the upcoming duration $p[n+1]$. Both RNNs learn to approximate the distribution over the next possible durations and pitches based on their inputs and their previous hidden state (recurrent connections). In order to generate melodies from the artificial composer, the upcoming duration and pitch are sampled from the estimated transition distributions (bars on the lower right).

and a set of fixed model parameters Θ. Then, the probability of a melody s could be expressed as the probability of all notes occurring in sequence:

$$P(x^s[1:N^s]) = \prod_{n=1}^{N^s} T\big(x^s[n] \mid \Phi^s[n], \Theta\big) \; . \tag{1}$$

However, the inputs and parameters of this distribution have to be precisely defined – they could in principle encompass many different dimensions: the notes, the harmony, the tempo, the dynamic, and meta-parameters like for example the composer's musical and nonmusical experiences, or his general state of mind at the time of composition.

A simple approximation of the distribution T is given by Markov chain models of order k [22], where the probability of the next note depends only on the model parameters Θ_{MC} and history dependent input consisting of the last k notes:

$$\Phi^s_{MC}[n+1] = x^s[n-k:n] \,. \tag{2}$$

Markov chain models, however, can capture only a fraction of possible long-range temporal structure inherent in music [15]: due to the fixed restriction on the amount of past events that can influence the probability of occurrence of a note, this class of models will not capture all the complexity and temporal dependencies of music even with parameters trained to give the best possible approximation (the maximum likelihood estimate for a given set of data).

Rich temporal structure can in principle be captured by models that rely on recurrent neural networks (RNN). RNNs are probabilistic models that have potentially unlimited memory since their internal (so called "hidden") state H_{RNN} can depend on the entire history of prior notes in the song s. In this way, the input for each note x at time n can depend on the network representation of the full history $H_{RNN}(x^s[1:n-1])$ in addition to the current note $x^s[n]$ and the set of parameters Θ_{RNN}:

$$\Phi^s_{RNN}[n+1] = H_{RNN}(x^s[1:n-1], \Theta_{RNN}) \cup x^s[n] \,. \tag{3}$$

Splitting the Joint Probability of Duration and Pitch. For the transition distribution $T(\Phi^s_{DAC}[n], \Theta_{DAC})$ of the DAC we chose a conditional approach, where the note transition distribution T is split into two transition distribution: T_D for *duration* transitions and T_P for *pitch* transitions.

This splitting of the transition distributions into pitch and duration is an important difference of this work to previous attempts at symbolic music modeling with RNNs [16–18]. By this splitting, the DAC explicitly encodes and learns the duration (and pitch) of each note. An important design choice is that we train two separate RNNs separately to approximate the two transition distributions while *providing the upcoming duration as an additional input to the pitch network*. This architecture conditions the distribution of upcoming pitches on the duration of the upcoming note, effectively allowing the DAC to take into account the relation between a note duration and its pitch. Conditioning the pitch predictions on the upcoming duration is motivated by the fact that in standard western music theory there exists a relation between the duration of a note and its pitch. For example, a short note followed by a longer note has a high chance to be at the end of a musical phrase: long notes act as a tension release, typically exemplified by a short leading-tone which resolves to a longer note a semitone higher or lower. Additionally, rhythm is generally a more stable feature of western music. Indeed, in a corpus of songs written in a common style, examples of similar rhythmical patterns are more frequent than examples of similar melodic patterns, i.e. the rhythm is more shared across songs than the melody. With a song's rhythm being less informative than its melody [23], we chose to condition the melody on the rhythm in order to optimize the *expressiveness* of

our model: for a given rhythmical (melodic) pattern, many (few) examples of different melodies will be seen by the model, from which the DAC could extract transition distributions.

By representing a note in song s by its duration and pitch as $\text{note}^s[n] = (p[n], d[n])$, we can formalize the splitting of the joint probability in the DAC model in the general formalism introduced above:

$$\begin{aligned} T\big(\text{note}^s[n] \mid \Phi^s_{AC}[n], \Theta_{AC}\big) &= T\big(p[n], d[n] \mid \Phi^s_{AC}[n], \Theta_{AC}\big) && (4)\\ &= T_D\big(d[n] \mid \Phi^s_D[n], \Theta_D\big) \\ &\quad \times T_P\big(p[n] \mid \Phi^s_P[n], \Theta_P\big)\ , && (5) \end{aligned}$$

where Θ_D and Θ_P are the parameters of the duration and pitch networks, respectively. The inputs are given by

$$\Phi^s_D[n+1] = H_D(d^s[1:n-1], p^s[1:n-1], \Theta_D) \cup d^s[n] \cup p^s[n]\ , \quad (6)$$

$$\Phi^s_P[n+1] = H_P(p^s[1:n-1], d^s[1:n], \Theta_D) \cup d^s[n+1] \cup p^s[n], \quad (7)$$

where H_D and H_P are the internal (hidden) network states of the duration and pitch networks, respectively.

Summarizing, the DAC is constituted of two RNNs with similar architecture (see Fig. 1**B**). On the one hand, the duration network is trained to approximate the probability distribution over possible upcoming durations given the *current pitch and duration* and the network internal representation H_D of previous notes. On the other hand, the pitch network is trained to approximate the probability distribution over possible upcoming pitches given the *duration of the upcoming note* in addition to the current pitch and the network internal representation H_P (which contains the current duration).

Network Architecture. The DAC model was implemented in `python` using the `theano` library [24]. The pitch and duration RNNs are each composed of a binary input layer, four recurrent layers each composed of 128 gated recurrent units (GRU) and an output layer (see Fig. 1B). GRUs are a recent variant of the popular LSTM unit [5] that is particularly suited for sequence learning tasks and requires fewer parameters at similar performance [7,25].

The input layers are one-to-one mappings of the inputs to binary input units, in agreement with the splitting of the conditional probability (see the previous section). The input to the duration network $\mathbf{x}_D[n]$ is a concatenation of the duration and pitch vectors $\mathbf{d}[n]$ and $\mathbf{p}[n]$. The input to the pitch network $\mathbf{x}_P[n]$ is a concatenation of the current pitch and *upcoming duration* vectors $\mathbf{p}[n]$ and $\mathbf{d}[n+1]$. Each layer is then fully connected to every downstream layer.

In contrast to [19], we also increased the number of hidden layers to 4, which enables us to learn more complex structure and longer timescale dependencies.

The update equations for the activations of the recurrent layer $\mathbf{h}^i[n]$ at time step n for layer $i \in \{1, 2, 3, 4\}$ is given by the update equations of GRUs [10]. They are provided here with an adaptation to our architecture:

$$\mathbf{h}^i[n] = \mathbf{z}^i[n] \odot \mathbf{h}^i[n-1] + (\mathbf{1} - \mathbf{z}^i[n]) \odot \widetilde{\mathbf{h}}^i[n] \,, \tag{8}$$

$$\widetilde{\mathbf{h}}^i[n] = \tanh\left(\mathbf{w}_{y^i h^i}\mathbf{y}^i[n] + \mathbf{r}^i[n] \odot \mathbf{w}_{h^i h^i}\mathbf{h}^i[n-1]\right) , \tag{9}$$

$$\mathbf{z}^i[n] = \sigma\left(\mathbf{w}_{y^i z^i}\mathbf{y}^i[n] + \mathbf{w}_{h^i z^i}\mathbf{h}^i[n-1] + \mathbf{b}_z^i\right) , \tag{10}$$

$$\mathbf{r}^i[n] = \sigma\left(\mathbf{w}_{y^i r^i}\mathbf{y}^i[n] + \mathbf{w}_{h^i r^i}\mathbf{h}^i[n-1] + \mathbf{b}_r^i\right) , \tag{11}$$

where $\mathbf{z}^i[n]$ and $\mathbf{r}^i[n]$ are the activations of the update and reset gates, respectively, $\mathbf{w}$ and $\mathbf{b}$ are the network parameters, $\sigma(x) = (1+\exp(x))^{-1}$ is the logistic sigmoid function, $\odot$ denotes the element-wise product and $\mathbf{y}^i$ is the feed-forward input to layer i, which consists of both the global inputs $\mathbf{x}[n]$ as well as hidden layer activations $\mathbf{h}^{j<i}[n]$.

The update equation for the output unit k activation $\mathbf{o}[n,k]$ at note n is

$$\mathbf{o}[n,k] = \Theta\big(\mathbf{w}_{y^o o}\mathbf{y}^o[n] + \mathbf{b}^o\big)[k] \,, \tag{12}$$

where $\mathbf{y}^o$ is the feed-forward input of the output layer, which consists of the global inputs $\mathbf{x}[n]$ and all hidden layer activations $\mathbf{h}^{\{1,2,3,4\}}[n]$, and $\Theta(\mathbf{x})[k] = \frac{e^{\mathbf{x}[k]}}{\sum_j e^{\mathbf{x}[j]}}$ is the Softmax function. The Softmax normalization ensures that the values of the output units sum to one, which allows us to interpret the output of the two RNNs as probability distributions. In particular,

$$\mathbf{o}_D[n,d] \approx P(d[n+1]) \,, \tag{13}$$

$$\mathbf{o}_P[n,p] \approx P(p[n+1]) \,. \tag{14}$$

Training. To train the DAC, it is presented with melody examples from a training corpus and its parameters are updated in order to produce a shape of the estimated transition distributions T_P and T_D closer to the transition distributions of the training data.

A training epoch consists of feeding both networks 100 randomly selected melodies from a fixed partition of 80% of the melodies in the corpus (training set). After a whole song has been fed to the model, the parameters of the pitch and duration networks are updated with the ADAM optimizer [26] in order to maximize the log likelihood of the networks given the song s:

$$\mathcal{L}^s(\theta) = \frac{1}{N_s - 1} \sum_{n=1}^{N_s - 1} \log\Big(P\big(x^s[n+1]\big)\Big) \,, \tag{15}$$

where $P\big(x^s[n+1]\big)$ is given by the value of the activation of output unit that corresponds to the effective upcoming duration or pitch. After each epoch, we evaluate the predictive performances – the percentage of correctly predicted upcoming pitch or duration – of the DAC on a random sample of 100 songs from the remaining 20% of the corpus (validation set), as well as 100 random

songs from the training set to monitor the DAC performance. In contrast to [19], where training was stopped earlier, here we introduce a new stopping criterion: training is stopped when the average accuracy of the predictions over one epoch reaches a steady state on both the training and validation sets.

Composition of Melodies. At any given stage during (and after) learning the DAC can compose melodies. To do so, we iteratively add notes to a growing melody by sampling notes from the learned note transition distributions, while providing the previously sampled note as an input to the network. We first sample a duration from the duration network, which is then used as an additional input to the pitch network, from which finally sample a pitch:

$$d[n+1] \sim \text{Multinomial}\big(\Lambda(\mathbf{o}_D[n], \mathrm{T})\big)\,, \tag{16}$$

$$p[n+1] \sim \text{Multinomial}\big(\Lambda(\mathbf{o}_P[n], \mathrm{T})\big)\,, \tag{17}$$

where T is the sampling temperature and $\Lambda(\mathbf{x}, \mathrm{T}) = \frac{\exp(\ln(\mathbf{x})/\mathrm{T})}{||\exp(\ln(\mathbf{x})/\mathrm{T})||_1}$. Note that a sampling temperature lower than 1.0 produces a more stereotypical distribution as the entropy is decreased, while a temperature higher than 1.0 will make the landscape of the predictions more flat as the temperature increases until reaching maximum entropy corresponding to the uniform distribution. A temperature of 1.0 corresponds to a sample from a multinomial distribution with the parameters given exactly by the outputs of the DAC.

3 Results

We trained and evaluated the Deep Artificial Composer (DAC, see Methods) network with a corpus comprising 2408 Irish folk melodies [27] and 585 Klezmer tunes [28]. The songs in the corpus contained on average 250 notes per melody for a total of 748'250 notes.

Training the DAC. The evolution of the accuracy of the DAC can be assessed by observing the shape of the output distributions of both networks at different training times (see Fig. 2). We observe an increase in the accuracy of the estimated distributions. Indeed, at the latest stages of training, both the duration and pitch network can relatively well predict the true upcoming pitches and durations even though the melody has never been by the DAC before. The predictive performance, or accuracy, of the DAC, corresponds to the percentage of transitions where the model gave the highest probability to the true upcoming event. On average, the accuracy of the trained DAC networks on melodies from the validation/training set is 50%/80%. The average predictive performance of the duration model is 80%/85%.

Rhythm is a rather stable feature of the musical corpora we trained on, i.e. typically several examples of the same rhythmical patterns are found in the corpus – in contrast to similar motifs of consecutive pitches which are much

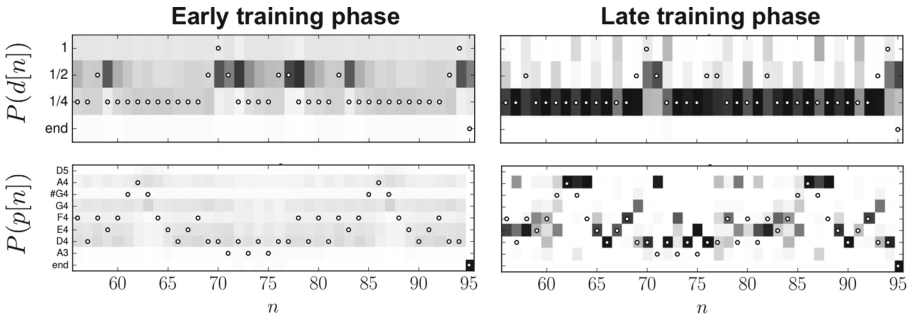

Fig. 2. DAC prediction performance during training: Evolution of the DAC predictions of duration (top) and pitch (bottom) during training from early (left) to late (right) training epochs. For each plot, the x-axis shows time steps, while the y-axis represents possible outputs of the networks. We show the last 40 predictions of the DAC on a randomly selected melody of the validation set. Small black and white circles represent the true upcoming notes, while shaded squares represent the probability distributions of the DAC's output units (white: $P = 0$, black: $P = 1$).

more variable. Thus, the duration network tends to show an overall higher accuracy. This observation further confirms our choice to condition the pitch transition distributions on the upcoming note duration, because by this the less accurate prediction is always conditioned by the more accurate one.

We invite the reader to convince himself of the increase in song structure as training progresses, by listening to `mp3` renderings of DAC-generated songs during different training phases at https://goo.gl/eUO9BR.

Assessing the Novelty of Melodies Generated by the DAC. In order to assess the quality of the generated melodies, we designed an automated measure of novelty, as an indirect measure of one feature of musical creativity [29]. For this, we define $\mathcal{N}_m^{\mathrm{C}}(s)$ as the novelty of a song s with respect to a corpus C and motif-size m. This consists in computing for the melody s which fraction of the song's transitions are *not found* in the musical corpus C, given only the last $m-1$ notes. If all possible transitions of a melody with preceding motifs of size $m-1$ can be found in the reference corpus, the novelty of the melody for the motif size m is 0.0. If all possible transitions with preceding motifs of size $m-1$ are absent from the reference corpus, the novelty of the corresponding melody is 1.0.

We applied this measure on the pitch and duration features independently for motifs of size 2, 4, 8 and 16, comparing sequences of the training set to songs in the validation set, to data generated by our model, and to sequences of pitches and durations generated by a Markov chain model of order 4 and 8 trained on the duration and pitch sequences independently (see Fig. 3**A**). An ideal artificial composer would produce melodies that show a novelty computed against the training set of similar magnitude as the validation set ($\mathcal{N}_m^{\mathrm{Train}}(\mathrm{Valid})$ on Fig. 3**A**), across all motif sizes. Melodies produced by the DAC indeed show

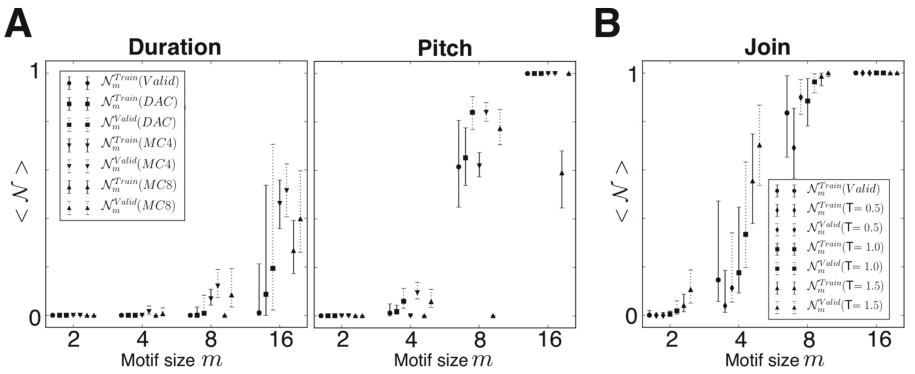

Fig. 3. Assessing DAC melodies by novelty measurement: The median, first and third quartiles of the novelty $\mathcal{N}$ of sequences (see text) with respect to the motif size m used for sequence comparison. (**A**) Novelty of sequences of pitches and durations separately (400 songs). Sampling temperature is 1.0. (**B**) Novelty of joint pitch and duration pairs for DAC songs generated at varying temperatures (400 songs each). The indicated corpora are: Valid = validation set, Train = training set, MC4 = Markov chain model of order 4 (400 songs), MC8 = Markov chain model of order 8 (400 songs).

a similar profile across motif sizes ($\mathcal{N}_m^{\text{Train}}$(DAC) on Fig. 3**A**), while Markov chain models tend to reproduce only motifs that they have already seen ("MC4" and "MC8" on Fig. 3**A**). This is especially true for motif sizes equivalent to the Markov chain order.

For both validation data and melodies generated by the DAC, the novelty profile gradually increases from 0.0 for motifs of size 2, up to completely novel choices of transitions for motifs of bigger sizes. For motifs of sizes bigger than 2, both validation melodies and sequences generated by the DAC are more novel in terms of their pitch sequence than their duration sequence. This further confirms the observation that rhythm is a stable feature of music that varies less across songs and motivate our choice to condition pitch predictions on duration (see Methods). In summary, the novelty profiles show that the DAC is able to create melodies that are *as novel with respect to the training data, as validation melodies from the same corpus*. However, because the DAC has to learn from examples, it generates melodies that are less novel with respect to songs in the training corpus than to songs from the validation set (compare $\mathcal{N}_m^{\text{Train}}$(DAC) to $\mathcal{N}_m^{\text{Valid}}$(DAC) in Fig. 3**A**).

The DAC not only generates songs that are closer to real songs (from the validation set) than Markov chain models in terms of novelty but by our design choice, it effectively models the relation between duration and pitch. To model the relation between duration and pitch with a Markov model, one would need to train it on the joint pitch and duration representation, which would yield to much sparser data and consequently even worse generative performance.

As a proxy for the final output of the model, we evaluate the novelty of joint sequences of notes (pitch-duration pairs) to study the effect of the sampling

temperature on the novelty of DAC generated melodies (Fig. 3**B**). We observe that the sampling temperature parameter can be used to coax the novelty of the artificial composer networks. Higher temperatures result in a more exploratory composer, while lower temperatures yield a less *inspired* artificial composer (as seen by less novelty in the generated sequences). Songs produced by the DAC with a sampling temperature of 1.0 are the closest to real data (from the validation set) in terms of novelty, which is expected since training is performed at this temperature.

Analysis of Melodies Produced by the DAC. The novelty profile across motif sizes introduced in the last section (cf. Fig. 3A) nicely aggregates how similar melodies are with respect to the given corpus – we can convert the novelty $\mathcal{N}$ to a similarity by calculating $1 - \mathcal{N}$. Therefore, we computed the novelty (similarity) profiles of generated joint melodies (pitch and duration pairs, motif sizes $2, 4, 8$, and 16) with respect to *only Irish melodies* and *only Klezmer melodies* of the training set, and used this data to train a simple classifier (3-Nearest Neighbor [30]) of melodic style.

The trained classifier can well distinguish between Irish and Klezmer melodies, given the similarity $(1 - \mathcal{N})$ profile of melodies with respect to the Irish and Klezmer reference corpora (see Fig. 4). Classification is almost always correct (error rate = 0.02) on data from the validation set. We then used the

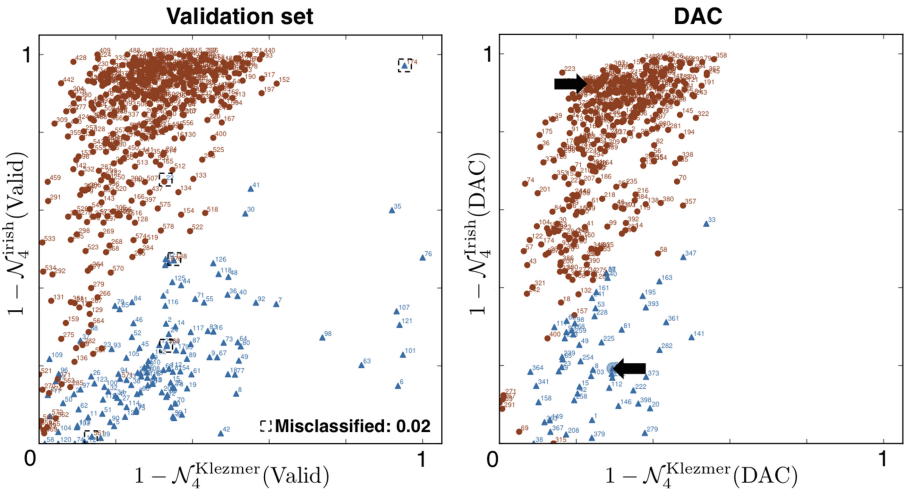

Fig. 4. Melody classification from the novelty profile: Similarities $(1 - \mathcal{N})$ of melodies from the validation set (left) and 400 DAC-generated songs (right) to either Klezmer melodies (x-axis) or Irish melodies (y-axis) of the training set. Similarity was calculated on motif-size $m = 4$. Each datapoint is labeled by the classification result (see text): circles for Irish and triangles for Klezmer. The few classification errors are highlighted with dashed squares. Sampling temperature for the generated melodies on the right figure is 1.0. Arrows point to the melodies displayed in Fig. 5.

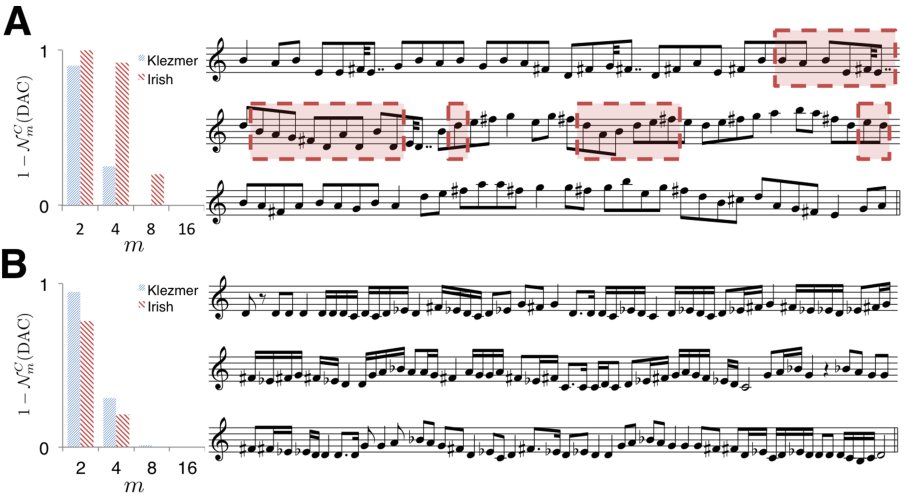

Fig. 5. Examples of DAC-generated melodies: Two melodies generated at sampling temperature 1.0. One melody classified as Irish (**A**) and one classified as Klezmer (**B**). Bar plots (left) show the similarity $(1 - \mathcal{N})$ of the song with respect to songs of the training set of a given style, across motif-sizes. Notes highlighted in red shaded areas correspond to transitions of the last note that are present in the training data given the last 7 notes (corresponding a motif-size 8 based novelty of 0).

trained classifier to label new melodies generated by the DAC. Comparing the two plots on Fig. 4, we observe that the distribution of songs with respect to their similarities is very resemblant between DAC-generated songs and songs of the validation set, validating the consistency of melodic styles generated by the DAC. It should be noted that DAC networks seem to produce songs with very high similarity to either dataset (close to the coordinate axes) not as frequently as the validation set, which is probably a remnant of their being underrepresented in the training set. For examples of labeled songs generated by the DAC networks, the reader is referred to `mp3` renderings at https://goo.gl/xTiYmg.

In Fig. 5, we present two example melodies generated by the DAC, that were classified as Irish or Klezmer (black arrows in Fig. 4) and their similarity profile $(1 - \mathcal{N})$. Both melodies are available as `mp3` renderings at https://goo.gl/r6WKHv. The Irish-classified melody (Fig. 5**A**) exhibits coherent rhythmical patterns across the tune. It seems to be in the 4/4 metric. The scale of the melody fits the one of the E Dorian traditional Gaelic music mode throughout the entire melody, and it ends and starts on the fundamental. The similarity profiles (left) show that for motifs of a size smaller than 16, the melody shows larger similarity to Irish reference motifs than Klezmer reference motifs. Interestingly, the ascendant movement that appears at the end of the melody is coherent with the first note, and would permit a repetition of the whole melody – while we did not explicitly encode this notion in the model, this seems to be a staple of Irish music and is often found in the Irish corpus that the model

was trained on. As such, it is also often present in DAC-generated melodies. Similarly, the melody labeled as Klezmer (Fig. 5**B**) shows coherent rhythmical patterns across the melody. The melody is in the scale of D Phrygian dominant traditional Klezmer mode and ends on the fundamental as well.

In summary, both example melodies imagined by the DAC present a very well-defined rhythmic and melodic structure that can be related to either Irish or Klezmer style. Notably, the modes of the two melodies fit either style. For both melodies, the artificial composer learned to end on the fundamental and display a well-defined structure, which can only be achieved if DAC networks were able to internally represent the mode through their hidden states.

4 Discussion

In this paper, we presented and evaluated the Deep Artificial Composer (DAC) – a model for the algorithmic composition of melodies. The use of a trainable architecture allows the model to automatically extract the invariant temporal dependencies specific to a given corpus of melodies. The DAC can, therefore, be trained on any monophonic music corpus in the `MIDI` format. In addition, we showed that DAC networks are able to learn similarities and divergences between different music styles, here on the example of Irish and Klezmer music. Indeed, most of the melodies generated by the trained artificial composer are consistent in style – when the model starts generating a melody with a structure close to an Irish folk song, the entire generated melody is close to the structure of an Irish tune. In similar ways, provided that it has been sufficiently trained, the artificial composer is also consistent in scale and rhythm.

Our model has to learn to correctly represent past events in order to be able to condition predictions on this internal representation of the history. By doing so, repetitions and more complex structures can be encoded and reused by the model during the generation of new melodies. However, probabilistic models are, by design, weak at repetition tasks: the only way for the model to exactly repeat the same sequence twice is to produce Dirac distributions. We have shown (see Fig. 2) that the DAC, as a probabilistic model, can come close to this. Additionally, because a sequence of durations is often redundant (durations are often repeated multiple times), the model has to learn an "internal clock" of the rhythm next to learning the transitions between different durations. We believe that a hybrid system with embedded grammars could improve the DAC, by externalizing some of the structure that is hard to produce with probabilistic networks. For example, learnable grammars such as the Sequitur algorithm [31] could be used to bias the network.

We demonstrated that the DAC as a model of melody composition produces melodies that are as similar to melodies in the training set as real tunes of the same style (from the validation set) are. To make this point, we introduced an indirect measure of creativity of generated sequences, by means of computing the novelty of produced sequences with respect to the trained corpus. Since the number of algorithms for automated music composition is increasing, there is

currently a lack of tools and methods for the automated evaluation of creativity of these systems [32,33]. While care should be taken to define the notion of quantitative *creativity* more carefully, our quantitative assessment of creativity without the intervention of human judges by the measure of novelty is, to our knowledge, the first of its kind. We would like to note that this method could be applied to any system generating (musical or nonmusical) sequences.

Finally, we used the novelty measurement to classify the style of melodies produced by the DAC. A much more sophisticated alternative would be to include such a classifier in the DAC itself by adding labeled *style* input and output units. These could act as latent variables that, after training, could be used to coax the DAC toward the generation of melodies in the corresponding style. Since such a classifier would be able to work on the full sequence history, we believe that the small number of misclassified generated melodies could be further reduced.

In order to illustrate this work, `mp3` examples generated by `GarageBand` from DAC-generated `MIDI` files are available for download at https://goo.gl/CHhoAu. Note that a clarinet is used for generated songs classified as Klezmer and a harp for generated songs classified as Irish.

Acknowledgments. The authors thank Samuel P. Muscinelli and Johanni Brea for their guidance and helpful comments. This research was partially supported by the Swiss National Science Foundation (200020_147200) and the European Research Council grant no. 268689 (MultiRules).

References

1. Seeger, C.: On dissonant counterpoint. Mod. Music **7**(4), 25–31 (1930)
2. Xenakis, I.: Formalized Music: Thought and Mathematics in Composition, vol. 6. Pendragon Press, New York (1992)
3. Lovelace, A.: Notes on l. menabrea's "sketch of the analytical engine invented by charles babbage, esq.". Taylor's Scientific Memoirs 3 (1843)
4. Fernández, J.D., Vico, F.: Ai methods in algorithmic composition: a comprehensive survey. J. Artif. Intell. Res. **48**, 513–582 (2013)
5. Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **6**(2), 107–116 (1998)
6. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with LSTM. Neural Comput. **12**(10), 2451–2471 (2000)
7. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint (2014). arXiv:1412.3555
8. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint (2013). arxiv:1308.0850
9. Karpathy, A.: The unreasonable effectiveness of recurrent neural networks. (2015). http://karpathy.github.io/2015/05/21/rnn-effectiveness/. Accessed 1 Apr 2016
10. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint (2014). arXiv:1406.1078

11. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
12. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
13. Gregor, K., Danihelka, I., Graves, A., Rezende, D.J., Wierstra, D.: Draw: a recurrent neural network for image generation. arXiv preprint (2015). arXiv:1502.04623
14. Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. arXiv preprint (2015). arXiv:1508.06576
15. Todd, P.M.: A connectionist approach to algorithmic composition. Comput. Music J. **13**(4), 27–43 (1989)
16. Boulanger-Lewandowski, N., Bengio, Y., Vincent, P.: Modeling temporal dependencies in high-dimensional sequences: application to polyphonic music generation and transcription. arXiv preprint (2012). arXiv:1206.6392
17. Franklin, J.A.: Computational models for learning pitch and duration using lstm recurrent neural networks. In: Proceedings of the Eighth International Conference on Music Perception and Cognition (ICMPC8), Adelaide, Australia, Causal Productions (2004)
18. Eck, D., Schmidhuber, J.: Finding temporal structure in music: blues improvisation with LSTM recurrent networks. In: Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing, pp. 747–756. IEEE (2002)
19. Colombo, F., Muscinelli, S.P., Seeholzer, A., Brea, J., Gerstner, W.: Algorithmic composition of melodies with deep recurrent neural networks. In: Proceedings of the First Conference on Computer Simulation of Musical Creativity (CSMC 2016), Huddersfield, UK (2016)
20. Walshaw, C.: abc notation. http://abcnotation.com/. Accessed 11 March 2016
21. Cuthbert, M., Ariza, C., Hogue, B., Oberholtzer, J.W.: music21, a toolkit for computer-aided musicology. http://web.mit.edu/music21/. Accessed 11 Nov 2016
22. Ames, C.: The markov process as a compositional model: a survey and tutorial. Leonardo **22**, 175–187 (1989)
23. Hébert, S., Peretz, I.: Recognition of music in long-term memory: are melodic and temporal patterns equal partners? Mem. Cogn. **25**(4), 518–533 (1997)
24. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: a CPU and GPU math expression compiler. In: Proceedings of the Python for Scientific Computing Conference (SciPy), vol. 4, p. 3. TX, Austin (2010)
25. Jozefowicz, R., Zaremba, W., Sutskever, I.: An empirical exploration of recurrent network architectures. In: Proceedings of the 32nd International Conference on Machine Learning (ICML 2015), pp. 2342–2350 (2015)
26. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint (2014). arXiv:1412.6980
27. Norbeck, H.: Henrik Norbecks's corpus of irish tunes. http://www.norbeck.nu/abc/. Accessed 03 Nov 2016
28. Chamber, J.: John chambers's corpus of klezmer tunes. http://trillian.mit.edu/jc/music/abc/Klezmer/. Accessed 03 Nov 2016
29. Gorney, E.: Dictionary of creativity: terms, concepts, theories and findings in creativity research (2007)
30. Fix, E., Hodges, J.L.: Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, DTIC Document (1951)

31. Nevill-Manning, C.G., Witten, I.H.: Identifying hierarchical strcture in sequences: a linear-time algorithm. J. Artif. Intell. Res. (JAIR) **7**, 67–82 (1997)
32. Jordanous, A.: A standardised procedure for evaluating creative systems: computational creativity evaluation based on what it is to be creative. Cogn. Comput. **4**(3), 246–279 (2012)
33. Loughran, R., ONeill, M.: Generative music evaluation: why do we limit to human? In: Proceedings of the first Conference on Computer Simulation of Musical Creativity (CSMC 2016), Huddersfield, UK (2016)