Music transcription modelling and composition using deep learning

Bob L. Sturm[1], João Felipe Santos[2], Oded Ben-Tal[3] and Iryna Korshunova[4⋆]

[1] Centre for Digital Music, Queen Mary University of London
[2] INRS-EMT, Montreal Canada
[3] Music Department, Kingston University, UK
[4] ELIS, Ghent University, Belgium

Abstract. We apply deep learning methods, specifically long short-term memory (LSTM) networks, to music transcription modelling and composition. We build and train LSTM networks using approximately 23,000 music transcriptions expressed with a high-level vocabulary (ABC notation), and use them to generate new transcriptions. Our practical aim is to create music transcription models useful in particular contexts of music composition. We present results from three perspectives: 1) at the population level, comparing descriptive statistics of the set of training transcriptions and generated transcriptions; 2) at the individual level, examining how a generated transcription reflects the conventions of a music practice in the training transcriptions (Celtic folk); 3) at the application level, using the system for idea generation in music composition. We make our datasets, software and sound examples open and available: `https://github.com/IraKorshunova/folk-rnn`.

Keywords: Deep learning, recurrent neural network, music modelling, algorithmic composition

1 Introduction

The application of artificial neural networks to music modelling, composition and sound synthesis is not new, e.g., [9, 17, 27, 37, 38]; but what is new is the unprecedented accessibility to resources: from computational power to data, from superior training methods to open and reproducible research. This accessibility is a major reason "deep learning" methods [8, 25] are advancing far beyond state of the art results in many applications of machine learning, for example, image content analysis [24], speech processing [19] and recognition [16], text translation [35], and, more creatively, artistic style transfer [13], and Google's Deep Dream.[5] As long as an application domain is "data rich," deep learning methods stand to make substantial contributions.

⋆ The authors would like to thank Dr. Nick Collins, Jeremy Keith creator and host of `thesession.org`, and its many contributors.

[5] `https://en.wikipedia.org/wiki/DeepDream`

Deep learning is now being applied to music data, from analysing and modelling the content of sound recordings [22,23,26,32–34,40,41], to generating new music [3,5,33]. Avenues for exploring these directions are open to many since powerful software tools are free and accessible, e.g., Theano [1], and compatible computer hardware, e.g., graphical processing units, is inexpensive. This has led to a variety of "garden shed experiments" described in a timely manner on various public web logs.[6] The work we describe here moves beyond our informal experiments[7] to make several contributions.

In particular, we build long short-term memory (LSTM) networks having three hidden layers of 512 LSTM blocks each, and train them using approximately 23,000 music transcriptions expressed with a textual vocabulary (ABC notation). We use this data because it is available, high-level with regards to the music it transcribes, and quite homogeneous with regards to the stylistic conventions of the music (it is crowd-sourced by musicians that play "session" music, e.g., Celtic, Morris, etc.). We take two approaches to training our models: one is character based, in which the system builds a model of joint probabilities of each textual character given the previous 50 characters; the other is "token" based, in which the system computes the joint probability of each token (which can be more than one character) given all previous tokens of a transcription. The result of training is a generative system that outputs transcriptions resembling those in the training material. Our practical aim is to create music transcription models that are useful in particular contexts of music composition, within and outside stylistic conventions particular to the training data.

In the next section, we review deep learning and LSTM, as well as past work applying such networks to music modelling and generation. Section 3 describes the specific models we build. In section 4, we analyse our generative models from three perspectives: 1) we compare the descriptive statistics of the set of training transcriptions and the generated transcriptions of a model; 2) we examine how a generated transcription reflects the conventions of a music practice in the training transcriptions (e.g., Celtic folk [18]); 3) we use a model for music composition outside the stylistic conventions of the training data. Our contributions include extending similar past work by using much larger networks and much more data (see Sec. 2.2), by studying the actual application of our models for assisting in music composition, and by making our datasets and software freely available.

2 Background

2.1 Long short term memory (LSTM) networks

A deep neural network is one that has more than one hidden layer of units (neurons) between its input and output layers [25]. Essentially, a neural network transforms an input by a series of cascaded non-linear operations. A recurrent

[6] deeplearning.net/tutorial/rnnrbm.html
www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks
www.wise.io/tech/asking-rnn-and-ltsm-what-would-mozart-write
elnn.snucse.org/sandbox/music-rnn

[7] highnoongmt.wordpress.com/2015/05/22/lisls-stis-recurrent-neural-networks-for-folk-music-generation

neural network (RNN) is any neural network possessing a directed connection from the output of at least one unit into the input of another unit located at a shallower layer than itself (closer to the input). A deep RNN is a stack of several RNN layers, where each hidden layer generates an output sequence that is then used as a sequential input for the deeper layer. With deeper architectures, one expects each layer of the network to be able to learn higher level representations of the input data and its short- and long-term relationships.

The recurrence (feedback) present in an RNN allows it to take into account its past inputs together with new inputs. Essentially, an RNN predicts a sequence of symbols given an input sequence. Training it entails modifying the parameters of its transformations to diminish its prediction error for a dataset of known sequences. The basic recurrent structure, however, presents problems related to exploding and vanishing gradients during the training procedure [20, 30], which can result in a lack of convergence of solutions. These problems can be circumvented by defining the hidden layer activation function in a smart way. One such approach defines long short term memory (LSTM) "cells", which increases the number of parameters to be estimated in training, but controls the flow of information in and out of each cell to greatly help with convergence [16, 21].

Though RNN and LSTM are not new, recent advances in efficient training algorithms and the prevalence of data have led to great success when they are applied to sequential data processing in many domains, e.g., continuous handwriting [15], speech recognition [16], and machine translation [35]. In the next subsection, we describe past applications of recurrent networks to music transcription modelling and generation.

2.2 Music modelling and generation using RNN and LSTM

Describing music as a sequence of symbols makes RNN immediately applicable to model it [3,4,10–12,17,29,36]. The RNN built and tested by Todd [36] consist of an input layer having 19 units, a single hidden layer with 8-15 units, and an output layer of 15 units. One unit in each of the input and output layers is the "note begin" state; 14 other units represent pitch, one each from D4 to C6 (no accidentals). Four other input units identify a specific monophonic training melody, of which there are four, each 34 notes long. Todd divides time such that each time-step of the model represents an eighth-note duration.

Mozer [29] builds RNN to model and generate melody using a distributed approach to music encoding. These systems generate output at the note level rather than at uniform time steps. Each pitch is encoded based on its fundamental frequency, chromatic class, and position in the circle of fifths. Note duration is encoded using a similar approach. Chordal accompaniment is encoded based on the pitches present. Some input units denote time signature, key, and downbeats. Mozer's RNN employs a single hidden layer with $\mathcal{O}(10)$ units. Training material include artificial sequences (scales, random walks), 10 melodies of J. S. Bach (up to 190 notes long), 25 European folk melodies, and 25 waltzes. Mozer finds these systems can succeed when it comes to modelling local characteristics of melody, e.g., stepwise motions, but fail to capture longer structures, e.g., phrasing, rhythm, resolution.

The finding of Mozer provided motivation for the work of Eck and Schmidhuber [11], the first to apply LSTM networks to music modelling and generation. Similar to Todd [36], they employ a local music encoding approach with 13 units representing 13 pitches (chromatic octave), and divide time using a minimum duration, e.g., sixteenth note. They also use 12 input units to designate pitches in an accompanying harmony. The hidden layer consists of two blocks of 8 LSTM cells each, with one block devoted to melody and the other to harmony. They make recurrent connections from the melody block to the harmony block, but not the other way around. They train the system on 6 minutes of 12-bar blues melodies with chord accompaniment, encoded at 8 time steps per bar. Each training song is 96 time steps long. Compared with the results of Mozer [29], Eck and Schmidhuber find that the LSTM network demonstrates an ability to model and reproduce long term conventions of this style. In a similar direction, Franklin [12] models jazz melodies and harmonic accompaniment using LSTM networks, but using a distributed music encoding similar to that used by Mozer [29].

Chen and Miikkulainen [4] "evolve" an RNN using fitness functions that quantify the success of a melody along different qualities, e.g., short-term movement, and pitch and rhythm diversity. They define some of these constraints to favor the melodic style of Bartok, e.g., pentatonic modes. Chen and Miikkulainen appear to encode a melody measure wise, using 16 pairs of pitch interval and duration. Output units are read in a linear fashion, with pairs of interval and duration, until the length of a full measure is completed.

Eck and Lapamle [10] applied LSTM networks to modelling long-term conventions of transcriptions of Irish folk music. Their music encoding divides time into eighth-note durations, with each note (between C3-C5) and chord getting its own bit. A novel aspect is that the LSTM network input is a linear combination of the current note and past notes from metrically related times, e.g., 4, 8, and 12 measures before. They train their systems on transcriptions of reels transposed to the same key: 56 from `http://thesession.org` (the source of our training data), and 435 from another database. They take care to reset the training error propagation at transcription boundaries.

More recently, Boulanger-Lewandowski et al. [2] apply RNN to modelling and generating polyphonic music transcriptions. They encode music by absolute pitch (88 notes from A0 to C8), quantised to the nearest quarter note duration. They train several networks on different datasets, e.g., Classical piano music, folk tunes, Bach chorals, and find the generated music lacks long-term structure. (We hear such results in the music produced in the links of footnote 4 above.)

3 Creating our generative LSTM networks

All our LSTM networks have the same architecture, but operate over different vocabularies and are trained differently. One kind we build, which we term *char-rnn*, operates over a vocabulary of single characters, and is trained on a continuous text file. The second kind we build, *folk-rnn*, operates over a vocabulary of transcription tokens, and is trained on single complete transcriptions. We next discuss our training data, and then the architecture and training of our systems, and finally how we use them to generate new transcriptions.

3.1 Music transcription data

Our transcription data comes from a weekly repository of `https://thesession.org/`,[8] an on-line platform for sharing and discussing music played in "traditional music sessions" (often Celtic and Morris). The collection does not include just music transcriptions, but also discussions, jokes, accompaniment suggestions, and so on. All transcriptions are expressed in "ABC" notation.[9] Entries in the repository look like the following real examples:

```
3038,3038,"A Cup Of Tea","reel","4/4","Amixolydian","|:eA (3AAA g2 fg |
eA (3AAA BGGf|eA (3AAA g2 fg|1afge d2 gf:|2afge d2 cd||
|:eaag efgf|eaag edBd|eaag efge|afge dgfg:|","2003-08-28 21:31:44","dafydd"
3038,21045,"A Cup Of Tea","reel","4/4","Adorian","eAAa ~g2fg|eA~A2 BGBd|
eA~A2 ~g2fg|1af (3gfe dG~G2:|2af (3gfe d2^cd||eaag efgf|
eaag ed (3Bcd|eaag efgb|af (3gfe d2^cd:|","2013-02-24 13:45:39",
"sebastian the megafrog"
```

An entry begins with two identifiers, followed by the title, tune type, meter, key, ABC code, date, and contributing user. Contributions vary in detail, with some being quite elaborate, e.g., specifying ornamentation, grace notes, slurs and chords. Most transcriptions are monophonic, but some do specify multiple voices. Many transcriptions have improper ABC formatting, are missing bar lines, have redundant accidentals, miscounted measures, and so on.

We create data for training our *char-rnn* model in the following way. We keep only five ABC fields (title, meter, key, unit note length, and transcription), and separate each contribution by a blank line. The two entries above thus become:

```
T: A Cup Of Tea
M: 4/4
L: 1/8
K: Amix
|:eA (3AAA g2 fg|eA (3AAA BGGf|eA (3AAA g2 fg|1afge d2 gf:|2afge d2 cd||
|:eaag efgf|eaag edBd|eaag efge|afge dgfg:|

T: A Cup Of Tea
M: 4/4
L: 1/8
K: Ador
eAAa ~g2fg|eA~A2 BGBd|eA~A2 ~g2fg|1af (3gfe dG~G2:|2af (3gfe d2^cd||
eaag efgf|eaag ed (3Bcd|eaag efgb|af (3gfe d2^cd:|
```

This leaves us with a text file having 13,515,723 characters in total, and 47,924 occurrences of `T:`.[10] There are 135 unique characters, e.g., "A", ":", and "^", each of which becomes an element of the vocabulary for our *char-rnn* model.

[8] `https://github.com/adactio/TheSession-data`

[9] `http://abcnotation.com/wiki/abc:standard:v2.1`

[10] This is not the number of transcriptions in the data because it also includes such things as user discussions and accompaniment suggestions for particular tunes.

We create data for training our *folk-rnn* model in the following way. We remove title fields and ornaments. We remove all transcriptions that have fewer than 7 measures when considering repetitions (to remove contributions that are not complete transcriptions, but transcriptions of suggested endings, variations, etc.). We remove all transcriptions that have more than one meter or key.[11] We transpose all remaining transcriptions (23,636) to a key with root C. All transcriptions are thus in one of the four modes (with percentage shown in parens): major (67%), minor (13%), dorian (12%), and mixolydian (8%). We impose a transcription token vocabulary — each token consists of one or more characters — for the following seven types (with examples in parens): meter ("`M:3/4`"), key ("`K:Cmaj`"), measure ("`:|`" and "`|1`"), pitch ("`C`" and "`^c'`"), grouping ("`(3`"), duration ("`2`" and "`/2`"), and transcription ("`<s>`" and "`<\s>`"). The two transcriptions above are thus expressed as

```
<s> M:4/4 K:Cmix |: g c (3 c c c b 2 a b | g c (3 c c c d B B a | g c (3
c c c b 2 a b |1 c' a b g f 2 b a :| |2 c' a b g f 2 e f |: g c' c' b g
a b a | g c' c' b g f d f | g c' c' b g a b g | c' a b g f b a b :| <\s>
<s> M:4/4 K:Cdor g c c c' b 2 a b | g c c 2 d B d f | g c c 2 b 2 a b |1
c' a (3 b a g f B B 2 :| |2 c' a (3 b a g f 2 =e f | g c' c' b g a b a | g
c' c' b g f (3 d e f | g c' c' b g a b d' | c' a (3 b a g f 2 =e f :| <\s>
```

Our dataset has 4,056,459 tokens, of which 2,816,498 are pitch, 602,673 are duration, and 520,290 are measure. A majority of the 23,636 transcriptions consists of 150 tokens or fewer; and 75% have no more than 190. There are 137 unique tokens, each of which becomes a vocabulary element for our *folk-rnn* model.

3.2 Architecture

Each LSTM network we build has three hidden layers with 512 LSTM blocks each, and a number of input and output units equal to the number of characters or tokens in its vocabulary. We encode our transcriptions in a local fashion, like in [11, 36], where each element in the vocabulary is mapped to an input and output unit. (This is also called "one-hot encoding".) The output of each network is a probability distribution over its vocabulary. The total number of parameters in our *char-rnn* model is 5,585,920; and that in our *folk-rnn* model is 5,621,722.

3.3 Training

We build and train our *char-rnn* model using the "char-rnn" implementation.[12] This employs the RMSprop algorithm[13] using minibatches of 50 samples containing 50 characters each, and a gradient clipping strategy to avoid the exploding

[11] By converting the remaining transcriptions to MIDI, we find the following: 78,338 measures of incorrect lengths (miscounting of notes, among 725,000+ measure symbols), 4,761 unpaired repeat signs, and 3,057 incorrect variant endings (misspecified repetitions). We do not attempt to correct these problems.

[12] `https://github.com/karpathy/char-rnn`

[13] T. Tieleman and G. Hinton, "Divide the gradient by a running average of its recent magnitude," lecture 6.5 of Coursera "Neural Networks for Machine Learning," 2012.

gradients problem in the LSTMs. We initialise the learning rate to 0.002, and apply a decay rate of 0.95 after the first 10 epochs. We build and train our *folk-rnn* model using our own implementation. This also employs the RMSprop algorithm, but with minibatches of 64 parsed transcriptions each. Since transcriptions in the dataset have different lengths (in number of tokens), we generate minibatches using a bucketing strategy, which places together in a minibatch sequences with approximately the same length, pads them to the maximum length using a "null" token, and then use a masking strategy to ignore null tokens when computing outputs and the loss function. We begin training with a learning rate of 0.003, and a rate decay of 0.97 applied after the 20 first epochs.

For both models, we clip gradients outside $[-5, 5]$ to the limits, and employ a dropout rate of 0.5 after each LSTM hidden layer. We train each model for 100 epochs in total. We use 95% of the dataset as training data and 5% as validation data (the latter for measuring progress in predicting characters or tokens). Through training, our *char-rnn* model learns a "language model" to produce ABC characters. On the contrary, our *folk-rnn* model learns a language model in a vocabulary more specific to transcription, i.e., a valid transcription begins with `<s>`, then a time signature token, a key token, and then a sequence of tokens from 4 types. Our *folk-rnn* model does not embody the ambiguity of meaning that *char-rnn* does, e.g., that `C` can mean a pitch, part of a pitch (`^C`), a letter in a title (`A Cup of Tea`), or part of a key designation (`K:Cmin`).

3.4 Generating transcriptions

With our trained models, it is a simple matter to have them generate output: we just sample from the probability distribution output by the model over its vocabulary, and use each selected vocabulary element as subsequent input. We can initialise the internal state of each model either randomly, or by inputing a valid "seed" sequence (e.g., beginning with `<s>`). Repeating the sampling process for N timesteps produces N characters/tokens in addition to the seed sequence.

4 Demonstrations of our generative LSTM networks

4.1 Statistical analysis of outputs

Comparing the descriptive statistics of system output with those of its training data is a straightforward way of assessing its internal model, but its relevance to the experience of music is highly questionable. We take our *folk-rnn* system and have it generate 6,101 full transcriptions. The proportions of meters and modes are close to those in the training dataset. Figure 1 shows the proportion of transcriptions of a particular token lengths, and the proportion ending with a particular pitch. The end pitch distributions appear to match between the two, but not transcription token length. We do not currently know the reason for this. We also find (by looking at the occurrence of repeat signs) that about 68% of the *folk-rnn* transcriptions use measure tokens creating a structure AABB with each section being 8 bars long; 54% of the transcriptions in the training data have this structure. This kind of structure is common in Irish folk music

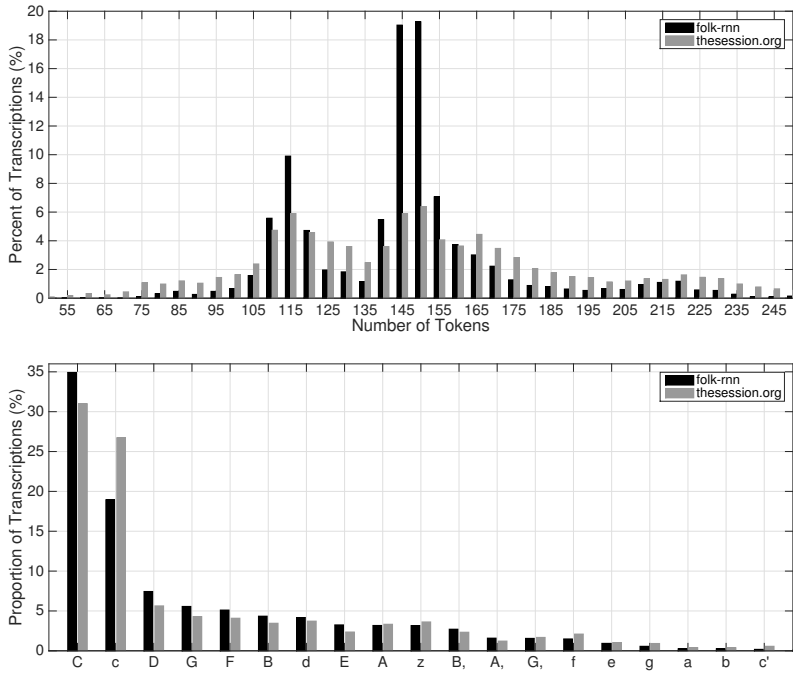


Fig. 1. Top: Distribution of the number of tokens in a transcription for the 6,101 transcriptions created by our *folk-rnn* system, compared with those in its (transposed) training dataset. Bottom: Proportion of transcriptions that conclude on a given pitch.

[18]. When it comes to errors, 16 generated transcriptions have the token `|1` (first ending) followed by `|1` instead of `|2`; and 6 have just `|1` or `|2` specified. Three transcriptions have incompletely specified chords, i.e., `]` appears without an accompanying `[`. (We corrected such problems when creating the training data for this model.)

4.2 Musical analysis of outputs

We generated 72,376 tune transcriptions from our *char-rnn* model, and automatically synthesised 35,809 of them (stopping only because of space limitations).[14] We used these results to create "The Endless Traditional Music Session,"[15] which cycles through the collection in sets of seven randomly selected transcriptions every five minutes. We shared this with the online community of `thesession.org`. One user listened to several, and identified the example below, saying, "In the tune below, the first two phrases are quite fun as a generative idea to 'human-compose' the rest of it! I know that's not quite the point of course. Still had

[14] We use `abc2midi` to convert each transcription to midi, and then process the midi using `python-midi` to humanise it for each of several randomly selected instruments, e.g., fiddle, box, guitar/banjo, and drums, and then use `timidity`, `sox` and `lame` to synthesise, master, and compress as mp3.

[15] `http://www.eecs.qmul.ac.uk/~sturm/research/RNNIrishTrad/index.html`

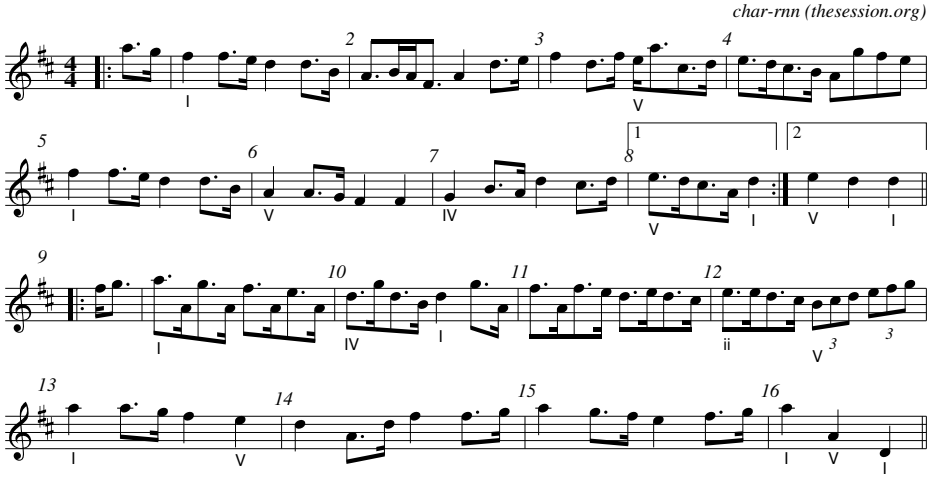


Fig. 2. Notation of "The Mal's Copporim," to which we add implied harmonies.

fun trying the opening of this one on the harp." Here is the exact output of our *char-rnn* model (notated in Fig. 2 with implied harmonies):[16]

```
T: Mal's Copporim, The
M: 4/4
L: 1/8
K: Dmaj
|: a>g | f2 f>e d2 d>B | A>BA<F A2 d>e | f2 d>f e<ac>d | e>dc>B Agfe |
f2 f>e d2 d>B | A2 A>G F2 F2 | G2 B>A d2 c>d |[1 e>dc>A d2 :|[2 e2 d2 d2 ||
|: f<g | a>Ag>A f>Ae>A | d>gd>B d2 g>A | f>Af>e d>ed>c | e>ed>c (3Bcd (3efg |
a2 a>g f2 e2 | d2 A>d f2 f>g | a2 g>f e2 f>g | a2 A2 D2 ||
```

Looking at this output as a composition teacher would the work of a student, we find no glaring mistakes: all measures have correct durations with accounting for the two pickup bars. Only the repeat sign at the beginning of the turn is unbalanced. We see that the piece is firmly in D major (but see discussion of harmony below), and each section ends with a resolution, the most strong being the conclusion. The melody appropriately slows down at these end points. The piece shows a structure very common to traditional Irish music [18]: a repeated 8 bar "tune" followed by a repeated 8 bar "turn." This is one point at which to suggest a change: just as for the tune, give the turn two endings, making the one already there the last, and compose a less conclusive resolution as the first.

Looking at the melodic characteristics in both the tune and turn, the dominant contour is the descent. The 3 stepwise notes beginning the piece, along with their rhythm, form a basic idea that is repeated verbatim or in transposition in several places. The piece shows clear use of repetition and variation: the turn keeps the dotted rhythm of the tune, but with a new melodic idea (for the first part of the phrase). The dotted rhythm is repeated often but also

[16] The system has in fact learned to create a title field for each transcription it produces because we include it in the training data for our *char-rnn* model.

varied. The occasional iamb adds variety and keeps the melody from becoming too monotonous, without breaking the strong metric character, but that idea is abandoned after the first 3 measures. While it serves well in m. 2&3, the iamb variety in the upbeat to the turn is less effective.

The tune and turn sound related, with the turn opening with a variation of the stepwise motion of the tune. Measures 9&10 in the turn vary bars 3 and 4 of the tune; and m. 13 in the turn recalls the beginning of the tune and its basic idea. Overall, the turn sounds rather aimless in its last half, and the giant leaps in the final bar are unexpected given the gradual motion in most of the piece. Here is a second point at which we can improve the composition: make bar 5 of the turn more closely related to its first bar, and change the rhythm of its second bar to that of the tune. The giant leaps in the last bar should be better prepared by the new first ending of the first suggestion above. Finally, in m. 6, change trochee rhythm to iamb and drop the second F-sharp to the D.[17]

The transcription may be monophonic, but harmony is implicit in the melody. (Chordal accompaniment became prevalent in session music since the early part of the 20th century [18].) In this piece, I (Dmajor) is the most common, (e.g., m. 1-3) with V (Amajor) appearing as well (e.g., m. 3&4), and IV (Gmajor) appearing in m. 10. There are some awkward harmonic moments: the V seems to arrive half a bar too early in m. 3; the first half of m. 10 is IV, but does one switch to V for the last beat, or keep IV and ignore the melodic A? The harmony in m. 12 could be ii (Eminor) — the only minor chord in the piece — which leaves m. 13 with a V-I cadence but to a weak beat. The second half of the turn is quite static harmonically, which contributes to its aimless quality. That is a third point where we can improve the composition.[18]

One might ask, in its generation of "The Mal's Copporim", whether the system is just reproducing portions of its training dataset. One characteristic element is the scalar run in the last half of m. 12. We find this appears 13 times in 9 training transcriptions, and in only three is it followed by the high A. Another characteristic pattern is m. 9, which appears (transposed) in only one training transcription,[19] but in the context of v (minor), and followed by a measure quite different from that in "The Mal's Copporim". Another characteristic element is the ending measure, which is not present in the training transcriptions. We find only one instance of m. 2,[20] but no instances of m. 3&4.

4.3 Music composition with the generative systems

We now describe an instance of using our *char-rnn* system to assist in the composition of a new piece of music. The process begins by seeding the system with the transcription of an idea, judging and selecting from its output, and seeding anew with an expanded transcription. We initialise the model with the following seed, which includes two bars:

[17] For example, `A>B A<G F2 D2`.

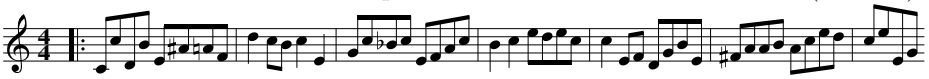
[18] One possibility is to change m. 13&14 to `a2 a>g f>A e>A | d2 A>d e2 f>g`.

[19] "Underwood" `https://thesession.org/tunes/5677`

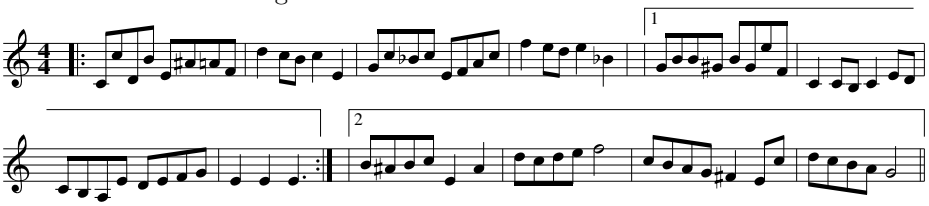
[20] Version 3 of "Durham Rangers" `https://thesession.org/tunes/3376`

```
T: Bob's Idea
M: 4/4
L: 1/8
K: Cmaj
|: CcDB E^A=AF | d2 cB c2 E2 |
```

It generates 1000 new characters, which include 18 measures following the seed to finish the tune. We notate a portion of this below with the seed (m. 1&2):



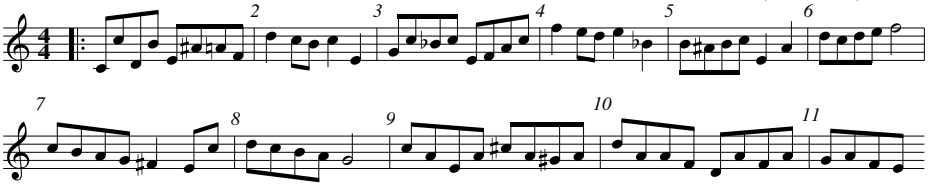
We keep the measure following the seed, compose another measure that varies the m. 2, and seed the system with those four measures. The system then produces two four-measure endings:



We keep the music of the second ending, and seed the system with

```
T: Bob's Idea
M: 4/4
L: 1/8
K: Cmaj
|: CcDB E^A=AF | d2 cB c2 E2 | Gc_Bc EFAc | f2 ed e2 _B2 |
B^ABc E2 A2 | dcde f4 | cBAG ^F2 Ec | dcBA G4 |
```

This produces 8 more measures, a few of which we notate below (m. 9-11):



We keep m. 9&10, vary them to create two new bars, then compose a few more measures to modulate to the V of V, and then repeat the first 15 measures transposed a whole step up. With a few more edits, we have composed "The March of Deep Learning", Fig. 3, which sounds quite different from the music in the training data transcriptions.

5 Discussion and reflection

The immediate practical aim of our work is to create music transcription models that facilitate music composition, both within and outside particular conventions. Toward this end, we have built two different kinds of generative systems using deep learning methods and a large number of textual transcriptions of folk

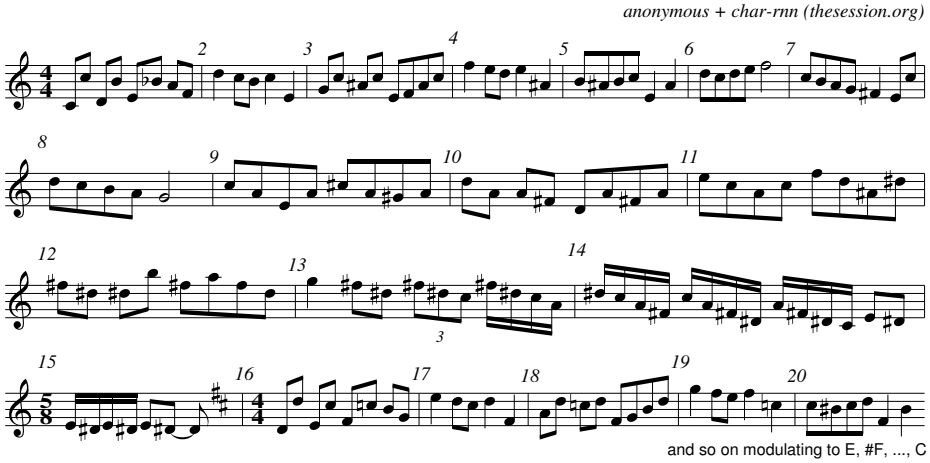


Fig. 3. The beginning of "The March of Deep Learning", composed with assistance from the *char-rnn* model, is quite different to the kind of music in the training data.

music, and demonstrated their utility from three perspectives. We compare the statistics of the generated output to those of the training material. We analyse a particular transcription generated by one of the systems (notated in Fig. 2) with respect to its merits and weaknesses as a composition, and how it uses conventions found in traditional Celtic music. We use one of the systems to help compose a new piece of music (notated in Fig. 3).[21]

The statistics of the output of the *folk-rnn* system suggest that it has learned to count, in terms of the number of notes per measure in the various meters present in the dataset. This is consistent with previous findings about RNN [14]. We can also see the distribution of pitches agree with that of the training data. The *folk-rnn* system seems to have learned about ending transcriptions on the tonic; and using measure tokens to create transcriptions with an AABB structure with each section being 8 measures long. In our latest experiments, we trained a *folk-rnn* system with transcriptions spelling out repeated measures (replacing each repeat sign with the repeated material). We find that many of the generated transcriptions (see Fig. 4) adhere closely to the AABB form, suggesting that this system is learning about repetition rather than where the repeat tokens occur.

A statistical perspective, however, is only able to reflect how well the learning algorithm has divined specific information about the training dataset to produce "valid" ABC output. To learn more specific information about how well these systems can facilitate music composition, we look at the level of individual transcriptions. We take on the role of a composition teacher assessing the work of a student. While the question of creativity and composition teaching is not without contention (for example, [28] and [7]), criteria such as creativity, imagination, originality and innovation are used in many music department when marking

[21] The reason why we use *folk-rnn* for the first part and not the others is purely because our preliminary experiments with LSTM networks involved *char-rnn*. Our results led us to refine the transcription vocabulary and training regimen for *folk-rnn*.

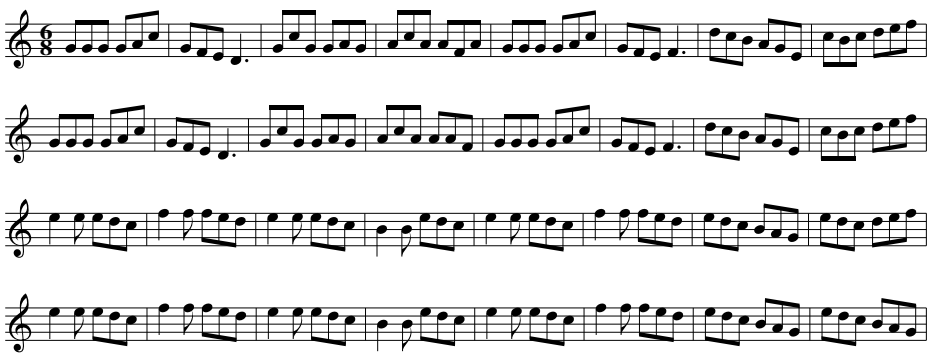


Fig. 4. Notated output of a *folk-rnn* model trained on transcriptions with repetitions made explicit.

composition assignments. Therefore, we can consider the perspective of a composition teacher a form of expert opinion on the ability of these systems, but be careful to acknowledge two things: 1) there is an inherited bias in Western musical culture with regards to the importance of the personal voice; 2) while stylistic awareness informs the discussion of a music composition, adherence to the conventions of a style is often not the primary focus of the ensuing discussion.

"The Mal's Copporim" (notated in Fig. 2), is a very plausible music transcription that is nearly "session-ready". Through our own audition of many hundreds of results, we also find others that have similar plausibility. Certainly, our systems produce many transcriptions that are much less plausible as well; and of course judging a transcription as plausible is, naturally, subjective; but the argument we are making here is that these systems are producing music transcriptions that appear to be musically meaningful (repetition, variation, melodic contour, structure, progression, resolution). We cannot dispense with the need for a curator to identify good and poor output; or for a composer/performer to correct or improve an output.

The role of the composer is clear when we apply our system to create a new piece of music in Sec. 4.3. Our intention behind seeding the system with the opening two bars of (Fig. 3) is to see how the system responds to an input that does not adhere closely to the stylistic conventions in its training data. Is it able to apply pattern variations even when the input pattern isn't very close to the learned material? Through our experience, we find the knowledge embedded within the system translates into this different context with the guiding hand of the composer. Within this relatively restricted approach to composition, we find our systems useful for assisting in music material generation that goes in directions we thought little to take.

Our work so far has merely examined the ability of these deep learning methods for modeling ABC transcriptions, but further work is clear. First, we will elicit discussions from `thesession.org` community about the transcriptions produced by *folk-rnn*, and how they can be improved with respect to stylistic and performance conventions. Hillhouse [18] mentions the openness of session mu-

sicians to incorporate new tunes into their performance repertoire, and so we are interested to see if any incorporate some of our results. Second, we will conduct interviews with session musicians to analyse *folk-rnn* transcriptions for their adherence to stylistic conventions, and how the experts would change the transcriptions to better fit the style. This will provide opportunities to improve the transcription model. Third, we will build an interface such that users can explore the system for composing new music (much the way we applied it in Sec. 4.3), and then measure how well it facilitates composition. We also seek ways to adapt the models to other kinds of stylistic conventions, and to analyse the significance of model parameters and network layers to the musical knowledge implicit in the dataset.

6 Conclusion

Facilitated both by the availability of data, and the excellent reproducibility of research in deep learning, our work extends past research in applying RNN and LSTM networks to music modeling and composition [4, 10–12, 29, 36] by virtue of size: whereas past work has used up to only a few hidden layers of a few dozen units, and a few hundreds of training examples, to generate only few example sequences, we have built networks containing thousands of units trained on tens of thousands of training examples, and generated tens of thousands of transcriptions. We explore the learned models in several ways. In addition to a comparison of the statistics of the generated transcriptions and the training data, we employ critical perspectives that are relevant to our aims: to create music transcription models that facilitate music composition, both within and outside particular conventions.

We make no claims that we are modelling music creativity [39]. As they stand, these models are black boxes containing an agent that uses probabilistic rules to arrange tokens [31]. Curation, composition and performance are required to make the generated transcriptions become music. However, at the level of the transcriptions, we find the collection of results to have a consistency in plausibility and meaningful variation. These LSTM networks are able to take a transcribed musical idea and transform it in meaningful ways. Furthermore, our models seem quite applicable in the context of traditional Celtic music practice because the creative practice of practitioners lies in their ability to arrive at novel recombinations of familiar elements [6]. Discovering a good balance between consistency and variation is part of the development of a composer's inner monitor and is a contributing factor to a composer's own style. That presents a unique point at which our system could positively contribute. However, it is still up to the composer to learn when and how to bend or break the rules to create music of lasting interest. The application of machine learning is no substitute.

References

1. J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: A CPU and GPU math expression compiler. In *Proc. Python for Scientific Computing Conf.*, June 2010.

2. N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proc. Int. Conf. Machine Learning*, 2012.
3. N. Boulanger-Lewandowski, G. J. Mysore, and M. Hoffman. Exploiting long-term temporal dependencies in NMF using recurrent neural networks with application to source separation. In *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, pages 6969–6973, May 2014.
4. C. J. Chen and R. Miikkulainen. Creating melodies with evolving recurrent neural networks. In *Proc. Int. Joint Conf. Neural Networks*, pages 2241–2246, 2001.
5. A. E. Coca, R. A. F. Romero, and L. Zhao. Generation of composed musical structures through recurrent neural networks based on chaotic inspiration. In *Int. Conf. Neural Networks*, pages 3220–3226, July 2011.
6. J. R. Cowdery. *The melodic tradition of Ireland*. Kent State Uni. Press, 1990.
7. C. Czernowin. Teaching that which is not yet there (stanford version). *Contemporary Music Review*, 31(4):283–289, 2012.
8. L. Deng and D. Yu. *Deep Learning: Methods and Applications*. Now Publishers, 2014.
9. M. Dolson. Machine tongues XII: Neural networks. *Computer Music J.*, 13(3):3–19, 1989.
10. D. Eck and J. Lapamle. Learning musical structure directly from sequences of music. Technical report, University of Montreal, 2008.
11. D. Eck and J. Schmidhuber. Learning the long-term structure of the blues. In *Proc. Int. Conf. on Artificial Neural Networks*, 2002.
12. J. A. Franklin. Recurrent neural networks for music computation. *J. Computing*, 18(3):321–338, 2006.
13. L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.
14. F. A. Gers and J. Schmidhuber. Recurrent nets that time and count. In *Proc. Int. Joint Conf. on Neural Networks*, 2000.
15. A. Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.
16. A. Graves, A.-R. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, pages 6645–6649, 2013.
17. N. Griffith and P. M Todd. *Musical networks: Parallel distributed perception and performance*. MIT Press, 1999.
18. A. N. Hillhouse. Tradition and innovation in Irish instrumental folk music. Master's thesis, The University of British Columbia, 2005.
19. G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and T. N. Sainath. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Process. Mag.*, 29(6):82–97, 2012.
20. S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116, April 1998.
21. S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. bibtex: Hochreiter1997.
22. E. Humphrey, J. P. Bello, and Y. LeCun. Feature learning and deep architectures: New directions for music informatics. *J. Intell. Info. Systems*, 41(3):461–481, 2013.
23. C. Kereliuk, B. L. Sturm, and J. Larsen. Deep learning and music adversaries. *IEEE Trans. Multimedia*, 17(11):2059–2071, Sep. 2015.

24. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, pages 1097–1105, 2012.
25. Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
26. H. Lee, Y. Largman, P. Pham, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Proc. Neural Info. Process. Systems*, pages 1096–1104, 2009.
27. M. Leman. Artificial neural networks in music research. In Marsden and Pople, editors, *Computer Representations and Models in Music*. Academic Press, 1992.
28. M. Lupton and C. Bruce. Craft, process and art: Teaching and learning music composition in higher education. *British J. Music Education*, 27(3):271–287.
29. M. C. Mozer. Neural network composition by prediction: Exploring the benefits of psychophysical constraints and multiscale processing. *Cog. Science*, 6(2&3):247–280, 1994.
30. R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *J. Machine Learning Res.*, 28(3):1310–1318, 2013.
31. J. Searle. Minds, brains and programs. *Behavioral & Brain Sci.*, 3(3):417–57, 1980.
32. S. Sigtia and S. Dixon. Improved music feature learning with deep neural networks. In *Proc. Int. Conf. Acoustics, Speech Signal Process.*, pages 6959–6963, May 2014.
33. A. Spiliopoulou and A. Storkey. Comparing probabilistic models for melodic sequences. In *Proc. Machine Learn. Knowledge Disc. Data.*, pages 289–304, 2011.
34. B. L. Sturm, C. Kereliuk, and A. Pikrakis. A closer look at deep learning neural networks with low-level spectral periodicity features. In *Proc. Int. Workshop on Cognitive Info. Process.*, pages 1–6, 2014.
35. I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proc. Neural Information Process. Systems*, pages 3104–3112, 2014.
36. P. M. Todd. A connectionist approach to algorithmic composition. *Computer Music J.*, 13(4):27–43, 1989.
37. P. M. Todd and G. D. Loy. *Music and connectionism*. The MIT Press, 1991.
38. D. Tudor. Neural network plus. *(music score)*, 1992.
39. G. A. Wiggins, M. T. Pearce, and D. Müllensiefen. Computational modelling of music cognition and musical creativity. In R. T. Dean, editor, *The Oxford Handbook of Computer Music*. Oxford University Press, 2009.
40. X. Yang, Q. Chen, S. Zhou, and X. Wang. Deep belief networks for automatic music genre classification. In *Proc. INTERSPEECH*, pages 2433–2436, 2011.
41. C. Zhang, G. Evangelopoulos, S. Voinea, L. Rosasco, and T. Poggio. A deep representation for invariance and music classification. In *Proc. Int. Conf. Acoustics, Speech Signal Process.*, pages 6984–6988, May 2014.