Generating Convincing Harmony Parts with Bidirectional Long Short-Term Memory Networks

Andrei Faitas
Department of Informatics
University of Oslo, Norway
andreif@ifi.uio.no

Synne Baumann
Department of Informatics
University of Oslo, Norway
synneeb@ifi.uio.no

Torgrim R. Næss
Department of Informatics
University of Oslo, Norway
torgrirn@ifi.uio.no

Jim Torresen
RITMO and Department of Informatics
University of Oslo, Norway
jimtoer@ifi.uio.no

Charles P. Martin
RITMO and Department of Informatics
University of Oslo, Norway
charlepm@ifi.uio.no

ABSTRACT

Generating convincing music via deep neural networks is a challenging problem that shows promise for many applications including interactive musical creation. One part of this challenge is the problem of generating convincing accompaniment parts to a given melody, as could be used in an automatic accompaniment system. Despite much progress in this area, systems that can automatically learn to generate interesting and harmonically plausible accompaniments remain somewhat elusive. In this paper we explore systems where a user provides a sequence of notes, and a neural network model responds with an accompanying sequence of equal length. We consider two popular sequence-to-sequence models; one featuring standard unidirectional long short-term memory (LSTM) architecture, and the other featuring bidirectional LSTM. These are evaluated and compared via a qualitative study that features 106 respondents listening to eight random samples from our set of generated music, as well as two human samples. From the results we see a preference for the sequences generated by the bidirectional model as well as an indication that these sequences sound more human.

Author Keywords

Music Generation, Sequence Learning, Recurrent Neural Networks

CCS Concepts

•**Applied computing** → **Sound and music computing;** *Performing arts;* •**Computing methodologies** → **Neural networks;**

1. INTRODUCTION

Consider the concept of a melody—a sequence of notes connected and spaced out over time by a set of rules. The particular rules are arbitrary, but clearly defined through centuries of musical history. From this concept of melody, the idea of harmony arises as separate melodies played simultaneously are attempted. This necessitated a set of rules, not only for notes played in sequence, but for notes played together. Earlier work with machine learning and music has shown that deep neural networks (DNNs) can indeed produce sequences of notes that in large part adhere to the melodic and harmonic rules of western music [7, 9] and recent work [22] has demonstrated the potential to recreate and vary musical phrases using a sequence-to-sequence architecture.

While automatic accompaniment systems are widely used (e.g., [27, 16]), these usually generate music from a score or rules of musical theory. We propose that DNNs could be used in such a system to *automatically* learn to gener-

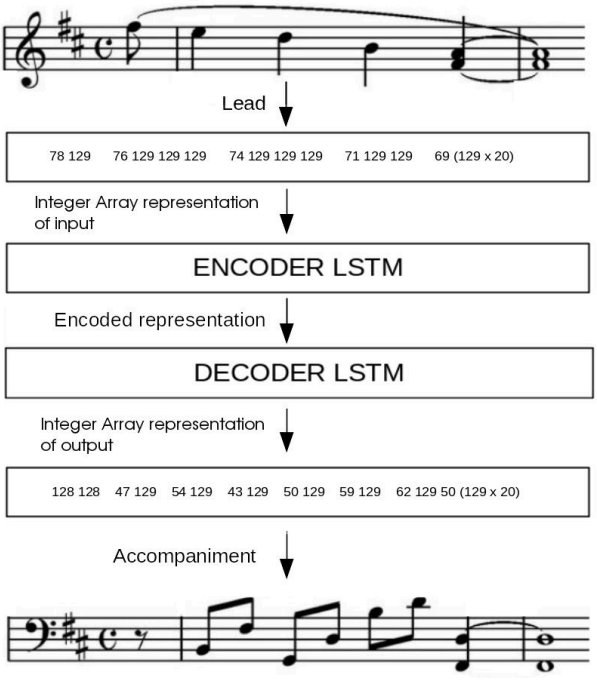

Figure 1: A musical sequence-to-sequence architecture to generate an accompaniment to a lead melody part. The lead is first converted an array with integer MIDI values (elaborated on in section 3.1). The encoder transforms the melody into a compact representation, which is used as input for the decoder to generate the integer array representation of the accompaniment.

NIME'19, June 3-6, 2019, Federal University of Rio Grande do Sul, Porto Alegre, Brazil.

ate accompanying melodies that will both respect the rules of harmony and sound convincingly musical. This process can be modelled as a sequence-to-sequence machine learning problem between matched examples of melodies and accompanying parts and can be considered to be similar to the problem of translating text between different languages [2, 24] where remarkable progress has been made using DNNs. In machine translation, one sequence is generated based on the other, which are both derived from a common set of symbols. We ask the following; can a neural network model be used to "translate" one sequence of notes into another in a way that makes them both harmonically compatible and convincingly human?

In this research, we propose and compare two sequence-to-sequence models to accomplish this task. Both models are recurrent neural networks (RNNs) applying long short-term memory (LSTM) cells [11], and both consist of two modules; an encoder network, that compresses the input melody into a low-dimensional representation, and a decoder, that stochastically generates an output melody using this representation as an input (see Figure 1). The models are distinguished by the structure of their decoder components: the unidirectional decoder can only view the musical sequence in a forward temporal direction (left-to-right in standard notation), the bidirectional network has modules that view the input sequence in both directions, combining this knowledge into the low-dimensional representation. Neither are novel models, thus showing the applicability of popular and established RNN designs for this task. For both models, our system takes a sequence of notes as input, and outputs a sequence of equal length where the notes (mostly) respect the rules for harmonies with respect to the input. The desired effect is that the input sequence can then be played simultaneously with the output sequence and be perceived as harmonically sensible by the human listener.

The main contributions of our work are implementations of these two models, and a strategy for data-augmentation of small music datasets, that can easily be integrated into interactive music tools, as well as a novel evaluation of these models through a qualitative study with 106 participants. This study sought to understand which model produces the most convincing results, and which generated accompaniments sound most human. From our results we conclude that there is a clear preference for the sequences generated by the bidirectional model, scoring well above its unidirectional counterpart in all three dimensions of assessment: overall preference, perceived degree of harmonic compatibility and whether the sequence seemed human-generated.

2. BACKGROUND

A melody is a sequence of notes; moreover, it is a sequence where the order of the elements is crucial for whether or not a human listener will consider it to be music or not. Machine learning models of melodies can learn to predict what note to add next based on both the current input and the previous predictions made, in order to achieve the musical coherency that a good ordering of the notes will provide. Markov models [1] have often been used to achieve this task, but recurrent neural networks (RNNs) are able to learn more distant dependencies between current and previous notes [20] and are a focus of much recent work in musical modelling. While other types of neural network (e.g., Music Transformer [12]) can be applied to music, the musical RNNs discussed in this research are accessible via frameworks such as Keras [3] for inclusion in interactive applications.

2.1 Recurrent Neural Networks

An RNN is a neural network designed to work with sequential data [3]. It achieves this by maintaining a hidden state that depends on both the input at the current time step, as well as the state given by the previous inputs. The effect is an internal memory; the network remembers inputs from the previous time steps. This means that a state at any given time step holds the information from each past time step, making it able to learn temporal, or sequential, structures.

2.2 Long Short-Term Memory

Although the RNN architecture can indeed predict sequences with internal dependencies with high accuracy, it is limited in regards to how long-term these dependencies can be. The cause for this limitation is linked to the problem of vanishing or exploding gradients; parameters are shared across time steps, so as the number of time steps increase the chain rule products in the gradient calculation become proportionately long, causing the gradients to either approach zero or grow exponentially [7]. A solution to this was proposed in 1997 by Hochreiter and Schmidhuber [11]. Dubbed long short-term memory (LSTM), their design mitigates the issue of the vanishing gradient by implementing a set of multiplicative gates to control the flow of information, and thereby increasing the prediction potential for longer sequences.

2.3 Bidirectional LSTM

The idea behind a bidirectional LSTM (BLSTM) [23] is to make use of all the information from both past and present in an input sequence by seeing it in both positive and negative time directions. This can be achieved by connecting two separate RNNs (one for each direction) to the same output layer. It has been shown that combining the idea of a bidirectional RNN with LSTM can give significantly improved results over unidirectional architectures in tasks where context is important [8]. In music, melodies and harmonies are affected by both previous and future events. This makes BLSTMs good candidates for music applications, such as generating chords to a monophonic melody [14].

2.4 Sequence to Sequence Learning

Sometimes it is desirable to have a network output a target sequence given a certain source sequence, such as in machine translation. One way to achieve this is with an RNN encoder-decoder architecture. This architecture consists of two separate recurrent neural nets; an encoder and a decoder, as illustrated in Figure 1. The idea is to encode a variable-length input sequence to a fixed-length state vector (denoted by "encoded representation" in Figure 1) that the decoder can use to extract the output sequence. The result is an output that is conditioned on the input sequence. This type of architecture has been shown to perform well in both translation tasks [2, 24] and event prediction [21].

2.5 Temperature

When constructing the output sequence, the softmax activation function is used to turn the scores for each possible output into a probability distribution. Before applying softmax in LSTMs (and neural networks in general) the balance of these scores can be scaled to control the randomness of the sampled output. This scaling value is commonly called "temperature".

The outputs of a neural network layer produce a logit vector $z = (z_1, \ldots, z_n)$. Given a temperature parameter T, the softmax procedure produces a probability vector $q =$

$(q_1, .., q_n)$ by comparing z_i with the other logits as follows:

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

The q_i parametrise a categorical probability distribution, which can be sampled to produce an index between 1 and n. With a temperature of 1, the softmax procedure is applied directly on the output of the previous layers. Through trial-and-error we chose to use a temperature parameter of 1.1 which results in slightly smaller values, and a softer probability distribution. This makes the model more sensitive to low probability candidates, resulting in more diversity—but also more mistakes.

For very high temperatures ($T \to \infty$), all predictions have very similar probabilities and the lower the temperature, the more the expected rewards affect the probability. With low temperatures ($T \to 0_+$), the probability of the prediction with the highest expected reward tends toward 1.

2.6 Neural Networks in Musical Interaction

Though many interactive music systems apply machine learning [18], the use of DNNs as part of an interactive system is rare. Vogl and Knee's intelligent drum machine [25] generated drum patterns using a DNN, and the Piano Genie [6] applies a DNN to compose piano music based on eight input keys. Data from touchscreens [19] and MIDI controllers [10] has been generated by RNNs, and a sequence varying RNN [22] has been integrated into a set of tools for the Ableton Live software. So far, the generation of novel, harmonically appropriate, accompanying sequences has not been incorporated into interactive tools. We propose that the sequence-to-sequence RNNs we compare would be appropriate for this application.

3. METHOD AND SYSTEM DESIGN

3.1 Data and Augmentation

As the basis for our dataset, we used 405 Bach chorales from Music21 [5]. From these 405 chorales, we separated out the four parts (soprano, alto, tenor and bass) as several of the scores included other solo instruments to accompany these.

Since the object was to train the RNN to learn to output a sequence that is harmonically compatible with the input, we trained the network on pairs of sequences. Each piece in our data set consist of four parts (or four sequences), soprano, alto, tenor and bass, so from each score we get two sequence pairs. We consistently paired up soprano with alto, and tenor with bass, where soprano and tenor were used as training input and alto and bass as targets.

We had two datasets: augmented and non-augmented. The non-augmented set consists of 810 sequence pairs taken from the chorales. The augmented set applied inversion and transposition to the parts of each chorale to produce a total of 4860 pairs. 10% of each set was reserved for testing. During training, each sequence was split into shorter sequences of 32 timesteps.

The chorales in our data set all have a certain structure; they all consist of four parts, and they are in the major or minor keys—meaning none of the samples are based on modal keys or are atonal or bitonal, nor do they have any complicated rhythmic features such as a polyrhythmic time signature. This is significant, mainly for the augmentation as this meant that inverting and transposing the parts would provide new sequences for the data set that were still valid for our training. We follow existing work [17, 15] to convert MIDI streams into arrays of integers with values between 0–129, as shown in Figure 2. The encoded interpretations are as follows:

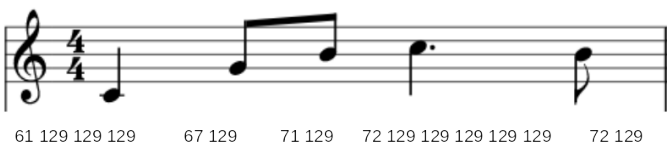

Figure 2: Integer representation of a monophonic melody. Values 0–127 represent MIDI "note on", 128-"note off", and 129-"no change". The first quarter note is represented by four integers where the first denotes the MIDI pitch and the next three indicate that there is no change for the next three semi-quavers.

- 0–127, play a note corresponding to that MIDI values
- 128, play nothing
- 129, play the same as previous element

This monophonic representation assumes a minimum time subdivision of 16th-notes (or semi-quaver), so that each number in the array represents a pitch to be played for one 16th of the time of a bar.

3.2 Implementation Details

Our basic model is very similar to the architecture illustrated in Figure 1. The encoder and decoder both have one layer of 256 hidden units, and there is a dense layer with the softmax activation function at the output of the decoder.

Generally, melodies are set in a sequential order, and the notes are affected by both forward and backward dependencies. They also often include periodic patterns which can be easier to predict when analyzed in both directions [4]. To explore the effect of analyzing sequences in both directions, we implemented the encoder as a bidirectional LSTM in one of our models. The forward and backwards state vectors from the encoder in this model are concatenated into a single vector of length 512, and the number of hidden units in the decoder is increased to 512 to account for this.

In a language model, sentences will usually be of different lengths, so one must use some form of strategy to tell the model when a sequence ends, such as end-of-sentence tokens or zero-padding to create fixed-length sequences. We made the assumption that since the target sequence is to be played alongside the input sequence, they must be of equal length. The decoder will then stop predicting after the generated output sequence has reached the same length as the input.

During training, the decoder learns to turn a target sequence into the same sequence, but offset one timestep forward using a method called teacher forcing [26]. Normally, the output at time t is fed into the network at time $t + 1$. With teacher forcing, the input at time $t + 1$ is instead replaced by the expected value from the training data.

We used a categorical cross-entropy loss function and ADAM optimizer [13] when training the models. The batch size was set to 128. The unidirectional model trained for 400 epochs on both datasets. The bidirectional model takes more time to train, so the final number of epochs was 248 for the non-augmented dataset and 200 for the augmented.

Both models start overfitting quite early, most noticeably in the bidirectional model. We attempted to combat this by using recurrent dropout with a fairly low probability of 0.2 in both the encoder and decoder network. This somewhat reduced the problem, but it also drastically increased the number of epochs required for the loss to converge. We decided to use the weights with the lowest training loss and disregard the increasing validation loss, as we evaluate the

model qualitatively and the results should sound pleasing to the human ear.

3.3 Qualitative study

Our goal was to compare the performance of unidirectional and bidirectional predictive models, and so we devised a qualitative study to achieve a measurable indication of their respective results. The study took the form of a questionnaire where participants revealed whether or not they actively played an instrument and if they had any training or education within musical theory.

The participants were then played 10 short examples of 2-part music (i.e., two sequences of notes were played simultaneously). Of these 10, there were 8 examples in which the second parts were generated by applying our models to randomly selected melody lines in the test sets, and the remaining 2 had both parts written by a human (i.e., the original matching part from J. S. Bach). For each example, the participants were:

- asked to rate how much they liked the music (scale 1 to 5)
- asked how well they thought the parts harmonized (scale 1 to 5)
- asked whether or not they thought the example was human made, or if one of the two parts were machine made.

A quantitative study for evaluating the performances was not devised, the reasoning for which lies in the observation that there are multitudes of possible sequences that could strictly be considered correct, (i.e., they are in the right key) but do not sound convincingly musical to the average human listener. Calculating the accuracy of these predictions is therefore not considered to be of interest.

The machine generated examples that the subjects listened to were created by taking the human-made melody sequence passed as input to the model, and the sequence generated as a result, and pairing them up. These were then played simultaneously creating a two part musical example, where the input melody sequence is considered part I and the generated output is considered part II. Of the 8 examples where part II was a sequence generated by the model, half were from the unidirectional model and the other half from the bidirectional model. Two of the examples from each model were sampled with a temperature value of 1.1, and the others with a value of 1.0. The temperature increases the randomness of predictions by a small amount, allowing the outputs to not always predict the notes with the highest probability.

4. RESULTS AND DISCUSSION

Since we want the model to learn the patterns and rules of music, and not necessarily create the same output as the targets, we focus on the human evaluation of the musicality of our results.

4.1 Comparison of individual sequences

The average rating of each sequence from our questionnaire are listed in Table 1, and a more comprehensive overview of the ratings of preference and musicality is shown in Figures 3 and 4.

Uni_Notemp_Norm: Unidirectional model without temperature and non-augmented dataset. This is one of the lowest rated sequences, but when examined by members of the authorship with formal musical backgrounds, this sequence is considered very strong. What makes the sequence

Data/model	Preference	Musicality	Human-like
uni_notemp_norm	2.59	2.44	16.81
bi_temp_norm	**3.27**	**3.32**	33.61
bi_temp_aug	3.11	3.17	35.29
bi_notemp_aug	2.91	2.89	31.09
uni_notemp_aug	2.66	2.82	33.61
uni_temp_norm	2.58	2.53	31.93
uni_temp_aug	*2.54*	*2.37*	*15.97*
bi_notemp_norm	3.04	3.07	**36.13**
Bach 1	3.20	3.62	36.97
Bach 2	3.96	4.04	63.87

Table 1: Mean ratings of each of our sequences from the qualitative evaluation of our model. The generated sequences are listed in the same order as they were presented to the critics. The best performing sequence in each category is marked in bold, and the worst in italics.

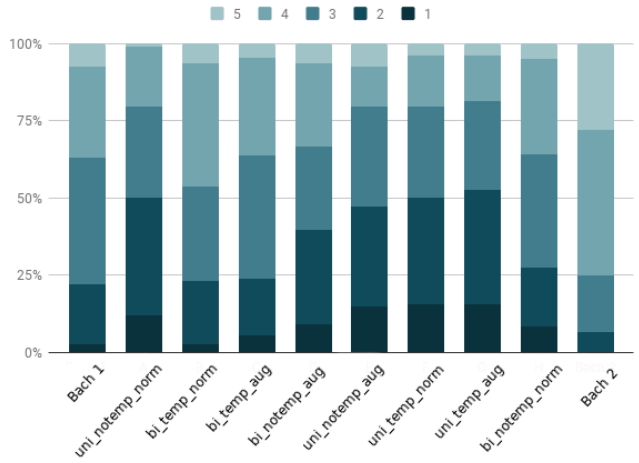

Figure 3: Stacked graph of the distribution of preference ratings on a scale from 1 to 5, 1 being the darkest blue and 5 the lightest. Bi_temp_norm and bi_temp_aug had the best rating for preference, suggesting that the bidirectional model with temperature sounded the most pleasant.

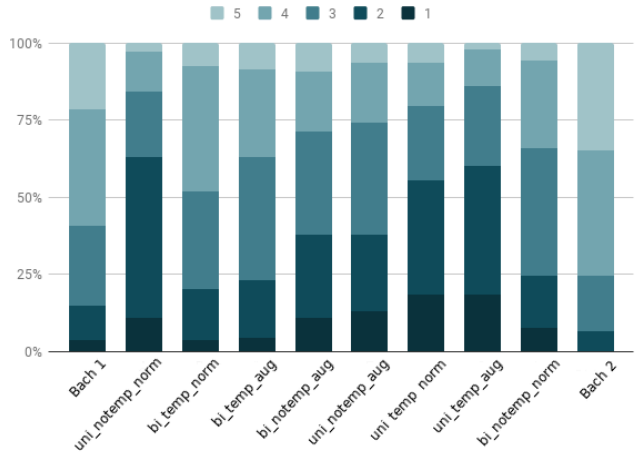

Figure 4: Stacked graph of the distribution of harmony ratings on a scale from 1 to 5. Bi_temp_norm and bi_temp_aug again show the highest scores, indicating that the bidirectional model with temperature is the most convincing in terms of harmony.

Human-like (%) rated overall and by trained critics		
Data/model	All	Musical training
Bidirectional	**34.03**	**37.50**
Unidirectional	*24.58*	*28.95*
Non-augmented data	29.62	30.92
Augmented data	28.99	35.53
Temperature	29.20	32.89
No temperature	29.41	33.55
Bach	50.42	50.00

Table 2: Table comparing the effects of the different changes to the model on how human-like the samples were rated, both overall and by the musically trained critics, with the best performing effect marked in bold. While non-augmented received a higher rating overall, the trained critics preferred the results with the augmented data.

fail is how little it seems to relate to the input sequence. The fact that it is this good on its own, but not so good in relation to the input, makes it seem like a consequence of over-fitting.

Bi_Temp_Norm: Bidirectional model with temperature and non-augmented dataset. This is the sequence that was rated highest in terms of preference and musicality, even rated higher on preference than one of our Bach-sequences.

Bi_Temp_Aug: Bidirectional model with temperature and augmented dataset. While being slightly outperformed by other results, this sequence has high ratings across the board, and demonstrates how proficient the bidirectional model is at generating a well structured harmony, even when trained on an augmented dataset.

Bi_Notemp_Aug: Bidirectional model without temperature and augmented dataset. This is the lowest rated bidirectional sequence. This is likely because of insufficient training time on augmented dataset for our bidirectional model and possibly an unlucky sample.

Uni_Notemp_Aug: Unidirectional model without temperature and augmented dataset. Some of the early predictions in the sequence sound out of key, and the same notes are often repeated throughout. The latter half achieves a very human-sounding rhythmic structure which we believe is the cause for it receiving a fair rating by respondents.

Uni_Temp_Norm: Unidirectional model with temperature and non-augmented dataset. Receiving average ratings on preference and musicality, this sequence sounds largely quite convincing. It suffers some decay during the latter half where many of the predictions are out of key. It sounds over-fitted on account of the very Bach-like middle part.

Uni_Temp_Aug: Unidirectional model with temperature and augmented dataset. By far the lowest rated sequence, and there are some notes here that sound unnatural to the average listener. It seems that while adding temperature or an augmented dataset works quite well with the unidirectional model, adding both makes it perform worse.

Bi_Notemp_Norm: Bidirectional model without temperature and non-augmented dataset. A fairly well rated sequence. It stays within the key throughout the sequence and maintains mostly pleasant-sounding harmonies with the input. The insistence on predicting off-beat rhythms may be why it is not rated higher as it creates a sense of disjointment from the input sequence.

4.2 The effects of individual changes

We would like to highlight the effect each of the changes we made to the model had on its performance. These are summarized in Table 2.

Data/model	Human-like (%)
Augmented / Temperature	25.63
Augmented / No temperature	32.35
Non-augmented / Temperature	32.77
Non-augmented / No temperature	26.47

Table 3: Comparative table of the effects of different combinations of datasets and temperature to the performance of the model. Adding either temperature or augmented dataset produced much better results than adding both or neither.

Bidirectional–Unidirectional: This has by far the biggest impact on the rating of every criteria. Changing from a unidirectional to a bidirectional model increases the average rating as human-composed by 9.45 (38.45%). We can see from Table 1 that most sequences generated by the bidirectional model were competitive with the first Bach sequence, which we consider a very positive outcome.

Augmented–Non-augmented: Overall this seemed to change little in terms of performance. Using the smaller, non-augmented dataset had a 0.63 (2.17%) increased rating as human-composed. However, as you can see in the right column in Table 2, this had more of an effect on the ratings of critics with formal musical training. Here the augmented dataset had the higher score, with a 4.61 (14.91%) increase over the non-augmented dataset.

Temperature–No temperature: Like the augmented versus non-augmented datasets, this change shows little effect on the rating of the sequences, with ratings of preference and musicality slightly favoring temperature (Fig 3 and 4), while no temperature was rated as slightly more human (with a 0.66 or 2.01% increase). It may then seem as if the choice of bidirectional or unidirectional model is the only change that has any significant effect on the ratings, however if we look at the different combinations of temperature and datasets from Table 3 we find some interesting results. Adding both temperature and an augmented dataset, or having neither produces relatively low ratings of "humanness" of 25.63 and 26.47 respectively, for an average of 26.05. Now, if we add only one or the other, the average rating increases significantly, with the addition of an augmented dataset increasing the rating by 6.30 (24.18%), and including temperature increases the rating by 6.72 (25.80%). This difference was even greater among musically trained critics. Naturally, the difference between these ratings are skewed by the poor performance of our unidirectional model with either both temperature and augmented dataset, or neither, but it is worth further exploring the effects of these changes with the bidirectional model.

5. CONCLUSIONS AND FUTURE WORK

From the results of our study it seems clear that the majority of our respondents preferred the music generated by the bidirectional LSTM model. We do note that there seems to be a higher level of acceptance for musical "weirdness" from the respondents that replied that they had some form of musical training; this seems to be reflected in the fact that these particular respondents believed the music to be human-generated more often than those without such training. There may be a need for a longer training period, particularly on the augmented dataset which could be addressed in future work. This makes us optimistic about the possible results for the bidirectional model, especially when looking at the second best sequence (sequence C). This se-

quence was ranked almost as highly as the highest ranked sequence even though its loss value had not had training time to reach a suitably low value.

From the responses we conclude that the bidirectional model is not only preferred by the majority of the 106 respondents, but also appears as the most significant factor in all three dimensions of assessment: preference, musicality and its similarity to man-made music. This leads us to recommend that bidirectional LSTM sequence-to-sequence models could be profitably implemented as part of interactive music systems, for instance, a predictive loop-based improvisation system, a topic that we intend to pursue in future work. While this work has focussed on accessible LSTM and BLSTM models, we could also explore emerging models such as musical variational autoencoders [22] or transformer networks [12] for this task. Finally, this research has shown that a musical sequence to sequence BLSTM can achieve high qualitative ratings even when a relatively small dataset is used for training.

Acknowledgments

This work was supported by the Research Council of Norway through its Centres of Excellence scheme (#262762) and the Engineering Predictability with Embodied Cognition (EPEC) project (#240862). It was also supported by by the Research Council of Norway and the Norwegian Centre for International Cooperation in Education as a part of the Collaboration on Intelligent Machines (COINMAC) project, under grant agreement (#261645).

6. REFERENCES

[1] C. Ames. The Markov process as a compositional model: A survey and tutorial. *Leonardo*, 22(2):175–187, 1989. doi:10.2307/1575226.

[2] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. EMNLP*, 2014. arXiv:1406.1078.

[3] F. Chollet. *Deep Learning with Python*. Manning Publications, Shelter Island, NY, 2018.

[4] Z. Cui, R. Ke, and Y. Wang. Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. In *Proc. UrbComp*, 2017. arXiv:1801.02143.

[5] M. S. Cuthbert and C. Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *Proc. ISMIR*, pages 637–642, 2010.

[6] C. Donahue, I. Simon, and S. Dieleman. Piano genie. In *Proc. IUI*, pages 160–164, 2019. doi:10.1145/3301275.3302288.

[7] D. Eck and J. Schmidhuber. Finding temporal structure in music: blues improvisation with LSTM recurrent networks. In *Proc. IEEE Workshop on Neural Networks for Signal Processing*, pages 747–756, 2002. doi:10.1109/NNSP.2002.1030094.

[8] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610, July 2005. doi:10.1016/j.neunet.2005.06.042.

[9] G. Hadjeres and F. Pachet. Deepbach: a steerable model for bach chorales generation. In *Proc. ICML*, 2017. arXiv:1612.01010.

[10] L. Hantrakul. GestureRNN: A neural gesture system for the Roli lightpad block. In *Proc. NIME*, pages 132–137, 2018.

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

[12] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. Music transformer: Generating music with long-term structure. In *Proc. ICLR*, 2019.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015. arXiv:1412.6980.

[14] H. Lim, S. Rhyu, and K. Lee. Chord generation from symbolic melody using BLSTM networks. In *Proc. ISMIR*, 2017. arXiv:1712.01011.

[15] Magenta. Melody RNN. Git Repository, 2016. URL: https://github.com/tensorflow/magenta/tree/master/magenta/models/melody_rnn.

[16] M. Marchini, F. Pachet, and B. Carré. Rethinking reflexive looper for structured pop music. In *Proc. NIME*, pages 139–144, Copenhagen, Denmark, 2017. Aalborg University Copenhagen.

[17] C. Martin. cpmpercussion/creative-prediction v1.0 [Git Repository], 2018. doi:10.5281/zenodo.1494040.

[18] C. P. Martin, K. O. Ellefsen, and J. Torresen. Deep predictive models in interactive music. *ArXiv ePrints*, 2018. arXiv:1801.10492.

[19] C. P. Martin and J. Torresen. RoboJam: A musical mixture density network for collaborative touchscreen interaction. In *Proc. EvoMUSART*, 2018. doi:10.1007/978-3-319-77583-8_11.

[20] M. C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 6(2-3):247–280, 1994.

[21] D. Q. Nguyen, D. Q. Nguyen, C. X. Chu, S. Thater, and M. Pinkal. Sequence to sequence learning for event prediction. In *Proc. of IJCNLP*, 2017. arXiv:1709.06033.

[22] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck. A hierarchical latent vector model for learning long-term structure in music. In *Proc. ICML*, 2018. arXiv:1803.05428.

[23] M. Schuster, K. K. Paliwal, and A. General. Bidirectional recurrent neural networks. *IEEE Trans. Signal Proces.*, 1997.

[24] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 2014. arXiv:1409.3215.

[25] R. Vogl and P. Knees. An intelligent drum machine for electronic dance music production and performance. In *Proc. NIME*, pages 251–256, 2017.

[26] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, June 1989. doi:10.1162/neco.1989.1.2.270.

[27] G. Xia and R. Dannenberg. Improvised duet interaction: Learning improvisation techniques for automatic accompaniment. In *Proc. NIME*, pages 110–114, 2017.