A Physical Intelligent Instrument using Recurrent Neural Networks

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

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

ABSTRACT

This paper describes a new intelligent interactive instrument, based on an embedded computing platform, where deep neural networks are applied to interactive music generation. Even though using neural networks for music composition is not uncommon, a lot of these models tend to not support any form of user interaction. We introduce a self-contained intelligent instrument using generative models, with support for real-time interaction where the user can adjust high-level parameters to modify the music generated by the instrument. We describe the technical details of our generative model and discuss the experience of using the system as part of musical performance.

Author Keywords

Embedded instruments, recurrent neural networks, generative models, interaction.

CCS Concepts

•Applied computing → Sound and music computing; •Computing methodologies → Neural networks; •Human-centered computing → *Interaction paradigms;*

1. INTRODUCTION

The use of machine learning algorithms to create musical compositions is a growing field of study. A lot of recent generative models applying deep neural networks, however, do not support direct human interaction, particularly in real-time during musical performance. Introducing ways for the user to manipulate the musical output will allow for such systems to be used in new kinds of intelligent interactive instruments, and to be explored through musical improvisation. Such instruments can be used by people with little or no previous experience to easily play around with musical ideas without the investment of time and money to learn a "real" instrument.

The main contribution of this research is a novel embedded device for interactively generating music with a recurrent neural network (RNN). This contrasts with many previous examples of music generation with neural networks that focus on offline, rather than interactive, generation of music, or require a powerful computer. Our device demonstrates that it is feasible to implement RNN-based music generation on a self-contained embedded platform, and explores manipulation of the continuous note sampling process as the main interactive function. This small device could be applied within many different musical scenarios and in setups with other instruments or equipment. This paper presents the design and implementation of an instrument prototype (Figure 1), and discusses the user experience along with some design considerations and ideas for future improvements.

Figure 1: A user interacting with our physical intelligent instrument: a device that continually generates and performs music using a recurrent neural network. The system is self-contained with controls for volume and sampling "temperature" as well as a built-in speaker for sonifying generated notes.

2. BACKGROUND

The use of artificial neural networks (ANNs) to create music and art has generated wide interest in recent years. ANNs can learn to transform any given picture to look like a painting created with different painting styles [9], and generate images that are practically indistinguishable from real photos [10]. ANNs can also be applied to synthesise musical audio waveform data [8], or translate music across musical instruments, genres, and styles [15].

Because neural networks can be trained to produce data learned from real-world examples, they can be good candidates for producing musical scores or performances without the need to program the rules of music theory manually. *Deep Artificial Composer* [5] can for example generate monophonic melodies resembling specified musical styles, and *Performance RNN* [17] can create polyphonic music with expressive timing and dynamics. Recently, interactive systems including ANN music generation have started to appear, e.g., *RoboJam* [13], a touchscreen music app that performs re-

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

sponses to short improvisations, and *Piano Genie* [7], which allows non-musicians to improvise on the piano. Neural nets can also be used to aid musicians in live performances, such as intelligent drum machines able to generate variants of rhythmic patterns provided as input by the user [19]. There is now potential to embed neural nets within smart musical instruments [18], or to create self-contained ANN music generators that could be used on stage or in the studio.

2.1 Artificial Neural Networks

ANNs were originally designed to imitate, in a simple way, the functionality of neurons in a real brain [4]. The basic building blocks are artificial neurons that take multiple input values, multiply them by respective weights and produce an output value. The input values can represent anything from pixels in an image to MIDI note values. Large numbers of these neurons are interconnected and arranged in a series of layers to form a network that can be trained for different applications such as pattern recognition and image classification. During training, the network is exposed to large numbers of examples with input values and expected output values. Based on the errors encountered in processing these examples, the network will adjust its weights to adapt to the training data.

2.1.1 Recurrent Neural Networks

Since music can be represented as a sequence of notes, the network must be able to predict this sequence based on both current and previous inputs; otherwise, the generated music will have no musical coherency. Recurrent neural networks (RNNs), are designed for the purpose of working with data sequences, and have been demonstrated to manage this task. The output of an RNN depends not only on its current inputs, but also on an internal state value that holds information from previous time steps, which allows it to learn to make decisions based on information it has seen in the past.

2.1.2 Long Short-Term Memory

Long short-term memory (LSTM) is a commonly used variant of RNNs that is able to capture much longer sequences than a regular simple RNN, which has a tendency to forget information when the sequences become longer. The LSTM achieves this by having an additional internal state called a "cell state" where information can be allowed to pass freely to later time steps, and by using a set of multiplicative gates to control the flow of information. There are three gates: an input gate, an output gate and a "forget gate". The input gate controls the information added to the cell state, making sure that irrelevant inputs do not pass through. Similarly, the output gate makes sure that irrelevant content is not passed along. The forget gate allows LSTM cells to reset when the content is no longer needed.

2.2 Music on Embedded Devices

Desktop computers and laptops are a well-established central component of digital musical instruments, but the use of single-board computers and embedded systems is becoming increasingly common. With the growing computing power and availability of single-board computers and microcontrollers, new platforms for creating digital musical instruments appear [14, 3]. Such platforms make it easy to prototype embedded instruments for performances and installations [16].

Self-contained, or embedded, instrument designs come with several advantages over the use of, for example, a laptop and simple microcontroller-based interface. The increased processing power of single-board computers allows more computationally intensive tasks to be performed natively than on a microcontroller, eliminating the need for external computers. Removing the use of general-purpose computers that are not dedicated to the instrument prototype can also increase longevity, as changes in other software might affect the functionality of the system [2]. The stability and portability of these systems suggest that they can be useful to artists who apply them within instrumental setups in live performance, or in their studios.

Figure 2: Close-up view of the physical intelligent instrument. The system has a built-in speaker and two knobs to control volume and sampling "temperature."

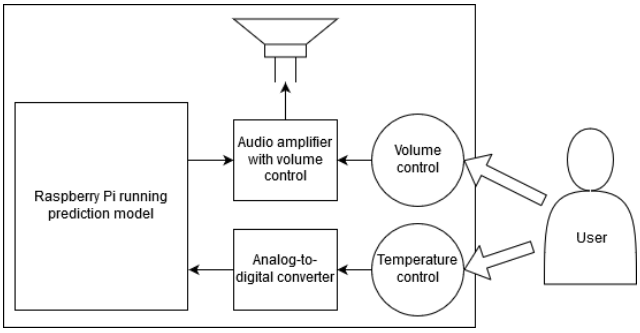

Figure 3: Diagram of the system. A generative music RNN model runs on a Raspberry Pi. The audio output goes through an amplifier with a potentiometer volume control and is played back on the built-in speaker. An analog-to-digital converter reads the voltage across a second potentiometer to control sampling temperature. The user can interact with the system by adjusting the two potentiometers.

3. DESIGN AND IMPLEMENTATION

Our physical intelligent instrument system, shown in Figure 2, consists of a box with a speaker and two knobs that the user can use to adjust certain parameters of musical predictions and playback. The system is designed to run music generation RNN models, and continually synthesise and sonify generated notes through an on-board speaker.

Figure 3 shows an overview of the system. The software runs on a Raspberry Pi Model 3 B+ single-board computer. An analog-to-digital converter (ADS1115) reads the voltage across a potentiometer to control the sampling temperature, and a 2.5 W mono audio amplifier (PAM8302), with another potentiometer for gain control, drives the speaker. The inside of the instrument is shown in Figure 4.

Figure 4: Hardware inside the enclosure. Hardware components consist of a Raspberry Pi, an analog-to-digital converter, two potentiometers, an audio amplifier and a speaker.

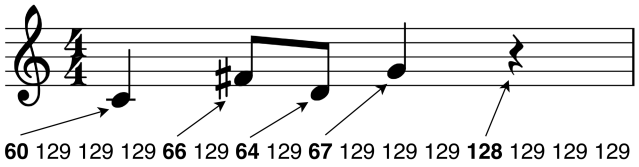

Figure 5: Notes are represented as integers with sixteenth note duration. 60, 64, 66 and 67 represent MIDI note numbers. 129 means no change, so the previous note will be held until a new note is played, or a value of 128 turns the note off.

3.1 Musical Representation

The system encodes music using sequences of integers in the range 0–129. 0–127 are pitches from the standard MIDI format, 128 tells the system to stop the note that was playing, and 129 represents no change. Each integer event has a duration of one sixteenth note. An example of the encoding is shown in Figure 5.

3.2 Network Architecture

The music generation RNN for the system is implemented in Python, using the Keras deep learning framework [4]. The RNN is in an auto-regression configuration, having been trained to predict the next in a sequence of notes. By returning the output of the RNN to the input, a continuous stream of musical notes can be generated.

The model consists of two LSTM layers with 256 cells each, an embedding layer on the input, and a dense layer with a softmax activation function on the output. The embedding layer transforms the input integer representation of notes into vectors of fixed size that the two LSTM layers can process hierarchically. The dense layer with softmax activation then projects the output from the LSTM layers back into probability distributions over possible note values. This style of neural network has been effectively used for creative sequence-learning tasks such as character-level text generation (CharRNN [11]), and music generation (MelodyRNN [1]).

3.3 Training

The training data is a set of 405 Bach chorales from the Music21 toolkit [6]. These chorales were split into four separate voices (soprano, alto, tenor and bass), and each voice was used as a single melody during training, giving a total of 1620 training examples of monophonic melodies. All melody lines were transposed to C major and A minor prior to training.

We trained the model for approximately 150 epochs with the Adam optimizer [12] and sparse categorical cross-entropy loss function, using a 90/10 training/validation split. Training was performed on an Nvidia GTX1070ti GPU. Due to the small size of the dataset, the model began to overfit before converging to a low loss value. It can be argued that this is not a big problem for this type of system, as the generated music should be perceived as pleasing to the human ear.

3.4 Sampling and Playback

When we sample from the model, it returns a vector of probability distributions over all possible note values. From this distribution, we can find the most probable next note based on previously predicted notes in the sequence. Two processes—one for sampling, and one for playback—run in parallel. The model is given a seed note as a starting point that it uses to predict the next notes in the sequence. After each prediction, the last note of the previous sequence is stored to be used as the seed for the next. This allows the model to play a continuous melody. The playback process sends notes over an internal MIDI connection for synthesis via the `Timidity++` software synthesiser and `FluidR3` soundfont. A shell script is used to ensure that the instrument processes are automatically started on boot. This, and the built-in speaker and controls, means that the instrument is completely self-contained and requires only USB power for performance.

3.5 User Interaction

After booting, the instrument will automatically start to play a continuous monophonic melody sampled from the LSTM network. By turning the two knobs on the instrument, the user can adjust two parameters of the system: volume and temperature. Temperature in this context is a hyperparameter that can be used to control the randomness of predictions by scaling outputs from the LSTM before applying softmax to calculate the probability distribution. A low temperature makes the model more conservative, making it less likely to sample from unlikely notes, while a higher temperature softens the probability distribution and allows the model to choose notes with a lower probability more often. Predictions with higher temperatures are more diverse, but can also have more mistakes. We use a temperature range of 0–10 in our model.

4. DISCUSSION

To give the feeling of real-time response from the instrument, we wanted to make sure the temperature adjustments begin to take effect as quickly as possible. Even though the system is fast enough to predict and play single notes, we found that predicting a sequence of two notes instead reduces the overhead without making any noticeable change in the response time from the temperature adjustment controls. This is not a big concern at this time, but might become more important when expanding the instrument with more functionality in the future.

The most important interaction in this context is the temperature control, which has a big impact on the generated melodies. At a temperature of 1, we can hear that the LSTM network has learned the structure and important musical elements of Bach chorales. When we increase the temperature to mid-range, the music begins to sound more like

experimental jazz, while still retaining some of the chorale elements. At higher temperatures, the music sounds like completely random notes, especially at max temperature where there is almost no coherency or structure. At the lowest temperature, the music tends to get stuck at a single note or pattern very quickly. We would argue that having the option to control the diversity of the generated melody results in a more enjoyable and rewarding user experience. There is another advantage of having a way to control the sampling temperature. When generating longer sequences, an LSTM network will often converge to a fixed state and become very repetitive. If this happens, the user can increase the temperature to make sure the model will include some more unlikely notes.

In our experience, the instrument is quite fun to use, and it grants the feeling of having some control over the music, even though it is generated by the LSTM model. However, more ways of interaction could be added in order to make this prototype a complete instrument, as it can become a little monotonous after a while. Possible extensions could be tempo control, choice of software synthesisers, or a MIDI interface where the user can connect a keyboard to play along with the instrument. It could also be useful to implement a way of choosing between multiple models trained on different data sets. If the users wants to train and test their own models, being able to switch between them would result in a more versatile instrument.

Experience with this system so far suggests that music generation RNNs could be a useful tool in music-making for both unskilled and expert users. Our system allows the sampling process to be explored in real-time during improvisation. As our system is self-contained, it can easily be integrated with other music-making devices, such as hardware synthesisers, or commercial music controllers.

5. CONCLUSIONS

In this paper, we have introduced a self-contained, physical, intelligent instrument that uses a deep recurrent neural network to generate music and supports real-time user interaction. Our work so far has demonstrated that real-time generation of music using an RNN is feasible on an embedded instrument using a Raspberry Pi. This system allows sampling diversity, or temperature, of the neural network model to be explored in real-time performance. Our initial explorations with the system suggest that this interaction, albeit simple, may help over some of the limitations of RNN music generation; in particular, the propensity for such models to become caught up in repetitive sequences. The fact that the RNN is made interactive may compensate for a model that is somewhat overtrained on a limited dataset. In future work we seek to understand how this system could form part of everyday music-making setups, and what kind of music it could afford or suggest. We are currently experimenting with an enhanced system that allows users to switch between different trained RNN models, between different software synthesisers, and to change tempo. This allows music from different styles or idioms to be generated. Code and schematics for both instrument versions is openly accessible on GitHub.[1]

6. ACKNOWLEDGMENTS

This work is supported by The Research Council of Norway as part of the Engineering Predictability with Embodied Cognition (EPEC) project #240862, the Collaboration on Intelligent Machines (COINMAC) project #261645, and the Centres of Excellence scheme, project #262762.

[1] https://github.com/edrukar/intelligent_instrument

7. REFERENCES

[1] D. Abolafia. A recurrent neural network music generation tutorial. [Magenta Project Blog Post], 2016.

[2] E. Berdahl. How to make embedded acoustic instruments. In *Proc. NIME '14*, pages 140–143, 2014.

[3] E. Berdahl and W. Ju. Satellite CCRMA: A musical interaction and sound synthesis platform. In *Proc. NIME '11*, pages 173–178, 2011.

[4] F. Chollet. *Deep learning with Python*. Manning Publications Co, 2018.

[5] F. Colombo, A. Seeholzer, and W. Gerstner. Deep artificial composer: A creative neural network model for automated melody generation. In *Computational Intelligence in Music, Sound, Art and Design*, pages 81–96. Springer, 2017.

[6] M. S. Cuthbert and C. Ariza. Music21: A toolkit for computer-aided musicology and symbolic music data. In *ISMIR*, pages 637–642. International Society for Music Information Retrieval, 2010.

[7] C. Donahue, I. Simon, and S. Dieleman. Piano genie. In *Proc. IUI*, 2019. doi:10.1145/3301275.3302288.

[8] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. In *Proc. ICML*, 2017.

[9] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proc. CVPR*, pages 2414–2423, 2016.

[10] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. In *Proc. ICML*, volume 37, pages 1462–1471, 2015.

[11] A. Karpathy. The unreasonable effectiveness of recurrent neural networks. Published on Andrej Karpathy's blog, May 2015.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR 2015*, 2015.

[13] C. P. Martin and J. Torresen. RoboJam: A musical mixture density network for collaborative touchscreen interaction. In *Computational Intelligence in Music, Sound, Art and Design: International Conference, EvoMUSART*, 2018. doi:10.1007/978-3-319-77583-8_11.

[14] A. McPherson and V. Zappi. An environment for submillisecond-latency audio and sensor processing on beaglebone black. In *Audio Engineering Society Convention 138*. Audio Engineering Society, 2015.

[15] N. Mor, L. Wolf, A. Polyak, and Y. Taigman. A universal music translation network. *ArXiV ePrint*, 2018. arXiv:1805.07848.

[16] V. E. G. Sanchez, A. Zelechowska, C. P. Martin, V. Johnson, K. A. V. Bjerkestrand, and A. R. Jensenius. Bela-based augmented acoustic guitars for inverse sonic microinteraction. In *Proc. NIME '18*, page 324–327, 2018.

[17] I. Simon and S. Oore. Performance rnn: Generating music with expressive timing and dynamics. [Magenta Project Blog Post], June 2017.

[18] L. Turchet. Smart musical instruments: Vision, design principles, and future directions. *IEEE Access*, 7:8944–8963, 2019.

[19] R. Vogl and P. Knees. An intelligent drum machine for electronic dance music production and performance. In *Proc. NIME '17*, pages 251–256, 2017.