GestureRNN:
A neural gesture system for the Roli Lightpad Block

Lamtharn Hantrakul*, Zachary Kondak
Georgia Tech Center for Music Technology
840 McMillan St. NW
Georgia Institute of Technology
Atlanta, GA 30318
lhantrakul3,zakondak@gatech.edu

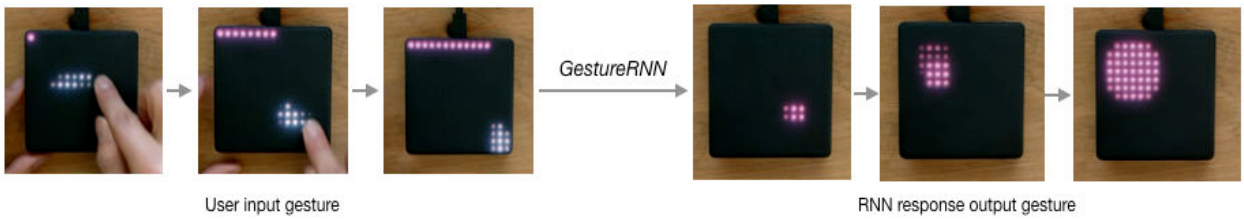

Figure 1. Generated sequences by the Recurrent Neural Network on the surface of the Roli Lightpad Block

**First author implemented deep learning and publication report. Second author contributed to the design and execution of the real-time server and Max/MSP patch.*

ABSTRACT
Machine learning and deep learning have recently made a large impact in the artistic community. In many of these applications however, the model is often used to render the high dimensional output directly e.g. every individual pixel in the final image. Humans arguably operate in much lower dimensional spaces during the creative process e.g. the broad movements of a brush. In this paper, we design a neural gesture system for music generation based around this concept. Instead of directly generating audio, we train a Long Short Term Memory (LSTM) recurrent neural network to generate instantaneous position and pressure on the Roli Lightpad instrument. These generated coordinates in turn, give rise to the sonic output defined in the synth engine. The system relies on learning these movements from a musician who has already developed a palette of musical gestures idiomatic to the Lightpad. Unlike many deep learning systems that render high dimensional output, our low-dimensional system can be run in real-time, enabling the first real time gestural duet of its kind between a player and a recurrent neural network on the Lightpad instrument.

Author Keywords
Recurrent Neural Networks, Gestures, Machine Learning, Deep Learning, Roli Lightpad Block

CCS Concepts
• Computing Methodologies → Machine Learning; **• Applied computing → Sound and music computing**; **• Hardware → Sensor and Actuators**

1. INTRODUCTION
The recent boom of machine learning (ML) and deep learning (DL) has greatly impacted the artistic community. In the visual arts, Deep Neural Networks have shown incredible capacity at generating artwork that can fool human beings [1], create new Japanese anime avatars [2] and hallucinate new Chinese characters [3]. In the musical domain, Eck et al [4] first showed how Long Short Term Memory (LSTM) networks can improvise Blues melodies, Bretan et al [5] used Recurrent Neural Networks (RNN) to generate music using a unit selection approach while Engel et al [6] employed a groundbreaking architecture for audio called *WaveNet* to interpolate between various different musical instrument sounds.

Whilst tools for using machine learning in artistic domains such as the popular *Wekinator* by Fiebrink et al [7] have been available for several years, it has only been in the last couple of years that newer methods such as deep learning, a subfield of machine learning, have been computationally feasible, open sourced and applicable to art. To put this intersection in context, The *Neural Information Processing Systems* or NIPS, a conference traditionally dedicated to statistical ML and DL breakthroughs, recently hosted a separate thread called "Machine Learning for Creativity and Design" in 2017.

However the majority of DL applications for art often focus on the final "high dimensional" output that is the finished piece of art itself. For example, a deep learning model generates the individual pixels of an image directly (64*64*3 ~ 4000 dimensions) [2] or generates the piano waveform directly (over 5M dimensions!) [8,10].

Arguably, humans do not operate at this level of high dimensionality. For example, an artist does not think about exact RGB pixel values but instead operates in broader brush strokes forming the final image. The movement of brushstrokes experienced by an artist is arguably in a lower dimensional space than the equivalent state space of pixel values seen by machine learning models. A musician does not improvise music by thinking of permutations of notes to fit on a final score. Instead, a combination of musical knowledge, the user's physical constraints, the instrument's physical constraints and current musical setting all influence the final choice of notes in a musical sequence [9].

We thus implemented a recurrent neural network (RNN) around the concept of this *low dimensionality* in the creative process. The system is designed to learn from a user's input in terms of three low dimensional values: instantaneous x, y and pressure p, which in turn give rise to the complex sonic output generated from Wekinator and Ableton Live. Our system, called *GestureRNN,* has no concept of the final sound, it only forms a representation of three input values

NIME'18, June 3-6, 2018, Blacksburg, Virginia, USA.

(x, y, p), and tries to predict and generate future triplets of (x, y, p). The system relies on learning these values from a musician who has already developed a palette of musical gestures idiomatic to the Lightpad, analogous to learning brushstrokes from an expert painter on a canvas.

This approach has several advantages: notably the deep learning system can be run in real time. Systems that render the final audio output can take hours to generate a short fragment of sound [10]. Moreover, the problem of learning and predicting lower dimensional values is arguably an easier task than modelling the high dimensional audio output directly. More importantly, this enables the player and *GestureRNN* to interact at the "gestural" level. Some of the sonic results from the interplay between model generated gestures and user input gestures result in surprisingly delightful timbres not possible with an equivalent duet between two physical hands on the Lightpad.

2. RELATED WORK

2.1 Overview

The notion of a system that "listens" and "completes" a player's musical input has been extensively explored. The most salient musical example is *The Continuator* by Pachet [11]. The system uses variable length Markov Models and a tree of patterns to form a statistical model of the user's MIDI input gathered from a Disklavier player piano. The system then generates a sequence that is melodically and rhythmically related to the original input, playing this back on the player piano.

In the deep learning domain, Bretan et al [12] have implemented real-time call and response systems using techniques such as unit selection and a *Deep Structured Semantic Model* [9]. The model is able to generate related musical sequences in real-time with a pianist.

GestureRNN on the other hand, operates in terms of predicting gestures, not the next discrete note. The system is closer to gestural systems that leverage controllers such as the Kaoss Pad [13], Microsoft Kinect [14], Leap Motion [15] or IMU gloves [16]. In these applications, parameters such as height or instantaneous coordinates are used to modulate sound generating parameters [13-16]. However, unlike these previous works, we present an interactive system in which machine-generated and expressive gestural responses are created, in real-time, by a *recurrent neural network* instead of Markov or probabilistic models.

GestureRNN was influenced by a generative system in the visual arts: a system called *SketchRNN* developed by Ha et al [17]. An encoder-decoder RNN was trained on thousands of human sketches and is able to "draw" new sketches based on tags such as a "firetruck" or "pig". The system can "complete" a seed sketch from a user by continuously regressing the change in x and y coordinates of the pen, as well as determine whether the pen is in a "lifted" or "writing" state. More impressively, *SketchRNN* can smoothly interpolate between contrasting input tags, such as generating a series of intermediate sketches between a "cow" and a "car" to produce a "cow-like car".

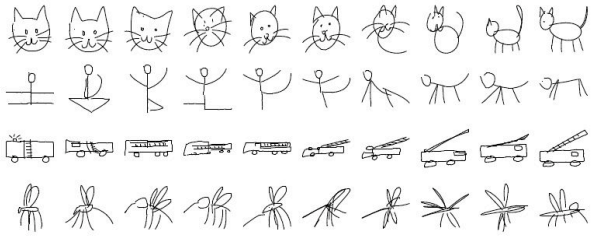

Figure 2: sketches from the *sketchRNN model*

2.2 Novelty

Analogous to learning the sketch movements as in Ha et al [17], we treat a musical gesture across the Lightpad Block as the "series of movements" to be learned.

Given a configuration of Wekinator's many-to-many mappings between the Lightpad's multi-polyphonic expression (MPE) output and an Ableton Live synthesizer, an expert player is able to generate many gestures that are idiomatic to the instrument. These include for example, rapid circular movements between zones on the Lightpad or straight lines that zigzag between different corners. It is precisely these idiomatic and expressive movements that the *GestureRNN* tries to learn and imitate. The "lower dimensional" generated gestures are then sent through the same Wekinator mapping to Ableton Live to produce the more complex and "higher dimensional" audio output.

The RNN essentially learns to generate musical output through the proxy of gesture. This approach enables the system to be run in real time in a live and improvisatory setting, a feat not currently possible with deep learning systems designed to render final output waveforms.

Moreover, unlike other XY pad-like controllers, the Lightpad features a programmable RGB LED panel below the sensor surface, enabling us to create an immersive experience by visually rendering *GestureRNN's* output directly on the instrument in tandem with sound. To our knowledge, this is the first implementation of a recurrent neural network used to generate gestures on an XY pad instrument like the Roli Lightpad. Moreover, the instrument's deformable tactile surface, full RGB LED matrix and our sound generation system all combine into a cohesive experience across modalities, facilitating fluid interactions between RNN-generated output and user gestures in a real-time, live performance setting.

3. IMPLEMENTATION

3.1 System Overview

A simplified system diagram of our system is shown in figure 3. Given the relatively young presence of deep learning tools, our implementation section is purposefully detailed to share best practices from machine learning literature with interested readers in the NIME community.

On a high level, the Lightpad Block sends (x, y, p) values to Max/MSP via the blocks max object provided by Roli. When a user plays a gesture on the Lightpad surface, Max/MSP forwards the triplet of values via OSC to the Wekinator. The three-dimensional values are interpolated into a 5 dimensional output from Wekinator. These correspond to 5 macro knobs in Ableton Live, which control a multitude of parameters in a synthesizer chain designed around the DAW's native *Tension* physical modelling synth. The final output sound is generated from Ableton Live.

At the same time, (x, y, p) triplets from the Lightpad are also forwarded to a python server running a *GestureRNN*. The predicted collection of values based off this seed gesture are sent back via OSC to Max/MSP, and in turn forwarded through the same aforementioned chain of Wekinator and Ableton Live to generate sound. Thus, it is possible for both *GestureRNN* and user to both be sending coordinates simultaneously.

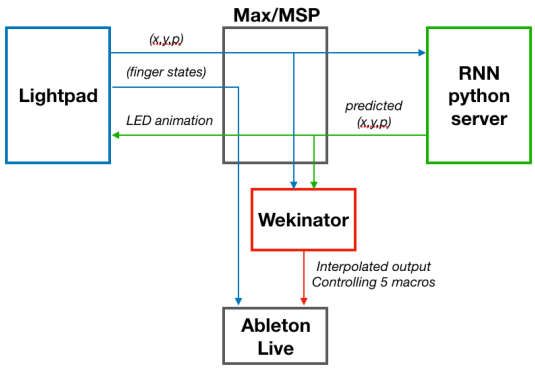

Figure 3: system diagram

3.2 Roli Lightpad Block

The Lightpad Block is a small and portable multidimensional polyphonic expression (MPE) instrument developed by Roli Inc. Readers may be familiar with the company's flagship product, the *Seaboard*, which features a unique rubber-like keyboard with multi-touch and support of five degrees of freedom per finger. The same technology is employed in the Lightpad Block but in a smaller, wireless and square form factor.

Figure 4: Roli Lightpad Block [20]

The device is primarily designed as an input controller to Roli's proprietary *Equator* AU/VST synth and *Noise* iOS application. These applications take full advantage of the Lightpad's MPE output in modulating sonic parameters. Roli also provide a well-supported blocks object for Max/MSP that make all instantaneous MPE values available to Max.

The Lightpad's RGB LED matrix enables the surface to be reconfigured into a clip launcher, drum pad, XY pad and a step sequencer. In this paper, we take advantage of the LED surface to render visual feedback given a user's input and render the behavior of the RNN.

3.3 Wekinator + Ableton Live

We designed a synthesizer chain in Ableton Live revolving around the DAW's native *Tension* physical modelling synthesizer. Five macro knobs are tied to synth parameters such as *force* and *position* as well as effect parameters such as *reverb mix* and *LFO rate* etc. The synthesizer chain is able to generate a wide range of timbres including metallic sounds, melodic sweeps and low rumbles. The reader is strongly encouraged to view the accompanying two-minute video [23]

The use of Wekinator enabled us to take advantage of Lightpad's MPE output and use the instrument beyond a simple XY pad. When the triplet of values from the Lightpad is sent into Wekinator, the interpolated output values enabled smooth and continuous control of the 5 macro knobs in Ableton Live, producing sounds rivalling in expressiveness with Roli's native *Equator* Synth.

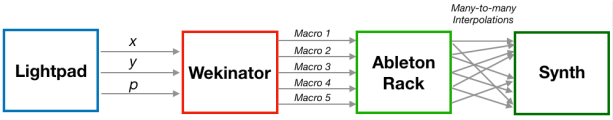

Figure 5: Mapping pipeline

Wekinator is "trained" by providing examples of an input and output state pair. For this application, we define four sonic states corresponding to each corner of the Lightpad. The top left corner for example, is a metallic shimmering sound while the bottom left has a smoother bass-like timbre. Although only four states were formerly defined in training, *Wekinator* uses neural networks to regress intermediate output values *between* these states. Thus when a user moves from a corner to the middle of the Lightpad, the changing input triplets are smoothly mapped to an interpolated output values representing varying mixtures of the four states. Wekinator excels at this task and enabled quick synthesizer prototyping in this manner.

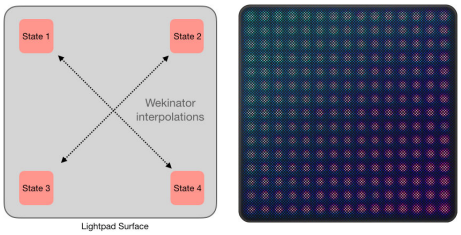

Figure 6: Synth engine mapping on Lightpad surface

3.4 GestureRNN training

A Long Short Term Memory (LSTM) is a type of recurrent neural network designed for time-based sequences. Unlike fully connected neural networks, such as the ones emplyed in Wekinator, an RNN receives two inputs: the input signal *and* the previous state of the RNN. This enables an RNN to keep a memory of earlier occurences in a sequence when making a prediction later in the sequence. A much more detailed treatment of RNN's and LSTM's can be found in existing machine learning literature [19]. The most important aspect of an LSTM is the presence of memory gates that control how much a previous state is used to influence the output of a current state. LSTM's have become foundational building blocks in language and time series modelling, leading to recent breakthroughs in machine translation [20] and predictive forecasting [21].

The LSTM architecture for *GestureRNN* is shown in figure 7 below. We used a simple two-layer LSTM with 64 and 32 hidden units respectively. These are followed by two layers of fully connected neurons (FC) of 16 and 3 units respectively. The last 3 units correspond to the regressed triplet of (x, y, p) predicted by the model.

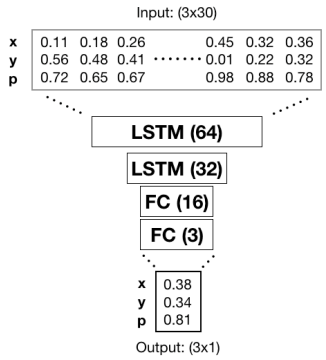

Figure 7: *GestureRNN* network architecture

The model is trained on 3000 datapoints sampled at a rate of 33Hz from an expert player of the Lightpad. We found this sampling rate to be optimal for the model to learn and generate smooth values across the Lightpad. We used a lookback length of 30 i.e. the LSTM sees 30 previous values (about 1 second) when making a prediction. We found that shorter lookback lengths did not enable the system to generalize and produce varied gestures. Thus the input to the model is (3x30) matrix consisting of 30 time-steps of (x, y, p) triplets. The output is a single (3x1) vector prediction corresponding to the next triplet (x, y, p). This is a regression task, meaning the network is trained to produce continuous values for each of the outputs (x, y, p). All input and output values are normalized between 0.0 and 1.0.

The network was trained for 100 epochs with a batchsize of 256, achieving an RMSE error of 0.004 and being able to generate sequences that mimicked the original training set. Figure 8 shows a graph of (x,y,p) triplets from the model. The system is implemented in Google's open sourced *tensorflow* framework and *keras* library.

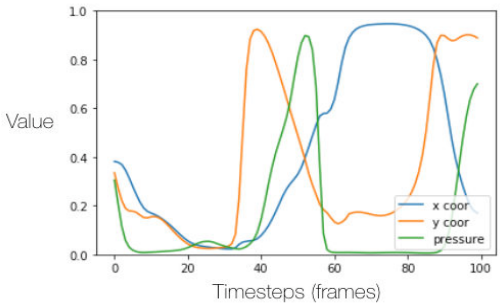

Figure 8: Continuous trajectories of (x, y, p) generated from *GestureRNN*

3.5 Real-time GestureRNN server

Although deep learning models take a long time to train, using the model for prediction or "inference" is a computationally cheaper task. Our model takes an average of 0.8 seconds to make a prediction on a CPU of a 2012 Macbook Pro, enabling the system to be used in real-time. The training process on CPU is approximately 20 minutes, but can be dramatically sped up using a GPU (Graphics Processing Unit).

As the user plays across the Lightpad, (x, y, p) triplets are sent to the python server via UDP and stored in a queue of values. Because the model was trained with a lookback of 30, it needs at least 30 (x, y, p) triplets or "frames" to make a prediction. When the server queue exceeds 30 frames, the model begins predicting a sequence.

We use a prediction length of 100 frames (4 seconds) i.e. the LSTM predicts 100 steps into the future after the input seed. We found that longer lengths caused the network to "even out". Past 3 seconds, the outputs would simply rest at the same values. This is largely because many of the original training gestures were about 3 – 5 seconds in length.

It is important to note the model does not directly output 100 frames. To generate 100 consecutive frames, we first feed the original seed window of 30 frames, which produces a single frame prediction. This single frame value is saved in an output buffer and at the same time, appended to the original seed window. The earliest value in the seed window is removed to maintain the length of 30 (like a FIFO queue of length 30). This new window is then passed into the LSTM to generate a second new frame. This second new frame is saved to the output buffer and then appended again to the seed window. This process of sliding the input window by appending the last predicted triplet is repeated iteratively until 100 frames are generated. It is standard practice in machine learning literature to predict values using LSTM's in this manner [22]. Figure 9 shows this process diagrammatically.

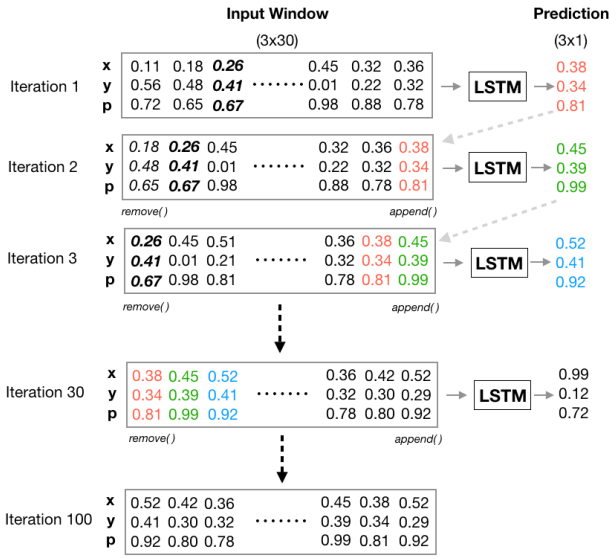

Figure 9: Iterative prediction using *GestureRNN*

Lastly, the output queue, now at length 100, is sent in bulk to Max/MSP, which steps through the predicted triplets at the original sampling rate of 33Hz so the sequence plays with correct timing.

3.6 Interaction considerations

To facilitate a robust and predictable real-time experience, we implement the following features:

- The user can activate *GestureRNN* by tapping three fingers simultaneously on the Lightpad surface. The RNN can be deactivated in the same manner
- When *GestureRNN* is activated, a progress bar is displayed on the topmost area of the Lightpad to signify how many frames have been captured. A full bar indicates the system has acquired the 30 samples it needs to make a prediction.
- If a user continues playing past the 30 samples, the server simply uses the most recent ones for prediction.
- The user can "interrupt" the *GestureRNN* mid-playback. This enables gestures to be chained and interrupted beyond a simple call-and-response paradigm with *GestureRNN*. By coupling longer gestures and strategically activating and deactivating *GestureRNN*, a fluent player can create long-term phrases.

4. PRELIMINARY EVALUATION

GestureRNN was demoed during an interactive session as part of an end-of-course demonstration. The authors acknowledge this does not constitute a comprehensive nor robust formal evaluation, which should be conducted in future work. This evaluation should be considered preliminary. Nonetheless, given the relatively small number of works leveraging RNN's and music hardware, the authors highlight how the observations noted here still provide insightful feedback on how the next iteration of *GestureRNN* can be improved as well as possible design paradigms that should be adopted when designing sequence generation models based on RNN's with a live user.

The interactive demonstration consisted of approximately 15 users; all were graduate students studying Music and nearly all have some musical training. Each user had approximately 5 – 10 minutes to interact with the system individually. Feedback was through the form of verbal interaction with no formal questionnaire.

4.1 Strengths

Many users opted for traditional call and response interactions, but when told that they could "interrupt" the generated sequence, began chaining together longer phrases. Longer term interactions between user and *GestureRNN* lasted approximately two to three minutes, indicating a degree of continuity in hand-off between the two agents relative to the short demo time each user spent. A full live performance would be better suited to evaluate the system's potential for phrase building in longer structures. A user study is appropriate, and will be implemented for the next iteration of *GestureRNN* with improvements based from this pilot interaction.

We note that many of the users exhibited some form of delight when *GestureRNN* was activated and played its first gesture across the surface. We noticed how users actively tried to understand and predict where the generated gestures would go next and carry on the gesture from there.

Users discovered how playing *on top* of the generated sequence resulted in synth timbres not possible with an equivalent duet of two fingers. To this extent, the system is very different from say, two musicians playing on the same guitar, or two musicians on two separate guitars. This is because one

cannot physically place two fingers on top of the exact same location on the instrument. However in this case, when a user presses in the same location as the generated sequence on the Lightpad, it is equivalent to two (x, y, p) triplets being sent simultaneously from the same location. The resulting synth texture is modified with a rapid LFO-like effect.

Many of the player remarked how they enjoyed the synthesizer sound design and how movements across the Lightpad generated surprisingly expressive sounds and changes in timbre. Many were surprise the synth engine was not developed directly by the manufacturer.

5.2 Weaknesses

The Lightpad surface is small (9cm x 9cm), limiting the range of motion available to both the user and the RNN. Moreover the pad nature of the Lightpad makes playing melodic sequences difficult. The recently released Lightpad "M" model fixes this problem by adding grooves on the surface of the instrument [18].

Many users attempted to play short percussive taps on the surface of the Lightpad. While the synth engine is designed to support these gestures, *GestureRNN* cannot generate successive taps in response to the user. This is perhaps the greatest pitfall of our current system.

To some users, the progress bar on top of the device was confusing. Some users kept playing continuously without ever lifting their finger, expecting the generated gesture to begin playing as they continued. In the current iteration of *GestureRNN*, a prediction is only made once the user ends a gesture i.e. lifts a finger. Approaches to account for other gestures and interactions are discussed in section 6: Future Work.

Some players wanted more unpredictability and did not appreciate how the *GestureRNN* could be turned "on" and "off". A secondary system that decides when to activate and deactivate may be an interesting avenue of research, but was beyond the scope of this paper.

5. FUTURE WORK

Our first implementation of an RNN for real-time gesture generation garnered positive feedback and excitement from a small group of pilot test users. To address some the current version's shortcomings in preparation for a larger user base, we propose the following changes.

- Multiple Lightpads can be chained together to produce a larger surface. Although it is possible to simply use a larger touchscreen, the user would lose the carefully engineered tactile properties of the Lightpad which nicely deform when pressed. A larger surface would enable broader strokes, as well as provide a larger area over which Wekinator can interpolate between the four sonic states.
- To address taps and gestures that momentarily lift off from the Lightpad surface, two important network architecture changes can be made. Firstly, we can adopt a training scheme more similar to *sketchRNN* in Ha et al [17]. Ha used a vector of 5 values $(\Delta x, \Delta y, p_1, p_2, p_3)$**.** Here, Δx and Δy refer to changes in position between frames rather than absolute coordinates. The first p_1 represents whether the pen is currently touching the paper. p_2 indicates whether the pen will be lifted after the current point. p_3 indicates whether the drawing has ended. Similarly, we can define a Lightpad gesture to have a similar feature vector of finger position $(\Delta x, \Delta y)$ and finger states (p_1, p_2, p_3), with the addition of $(\Delta pressure)$.
- Secondly, we can use what is known as a *Bidirectional Sequence to Sequence Variational Autoencoder* (VAE) [17]. Without going into too much detail, the network architecture has increased capacity to predict not only x and y positions, but the sequence of pen states that result in the final sketch. Like the original *sketchRNN*, this would enable the model to produce predictions of *variable length*, expanding the model's gestural vocabulary to shorter taps and longer arcs. The additional constraint of a normal distribution during training forces the network to output smooth transitions between predictions. This would enable the model to interpolate between various gesture trajectories and in theory, mix between its existing library and newly collected gestures from the current player.
- Although the system currently has no concept of the final sound output, this can be changed by extracting features such as MFCC's or a compressed STFT and feeding this into the model as a set of features in addition to the regular triplet of values. The size of the input feature vector needs to be kept at minimum, but this would enable the model to make predictions based not only on the instantaneous position and pressure, but also the timbre content of the current sound. The authors note how in reality, the feedback modality of timbre is an important part of how a gesture is created and thus, may be advantageous to model from a machine learning perspective.
- Lastly, we noted how the user was in charge of activating and deactivating *GestureRNN*. A more interesting interaction is allowing the system to decide when to come in. This can be achieved by training the model on duets between an expert player with the current system. By recording when the system is turned on and off, the coordinates of the player and sequence generated by the model, the system can learn to triangulate the best moment to enter the musical conversation.

6. CONCLUSION

In this paper we present *GestureRNN*, a neural gesture system for the Roli Lightpad Block. The system is modelled around the concept of using machine learning to operate in lower dimensional spaces associated with the creative process rather than the final high dimensional artistic output. It consists of two main components, *GestureRNN* and an accompanying synth engine. We trained an LSTM network to generate a sequence of movements based on collected data of an expert player. Our choice of network architecture and implementation enables the system to be run in real-time, allowing players to interact in a live improvisatory setting with *GestureRNN*. The results are duets between a player and RNN at the gestural level, giving rise to interesting and complex sonic output. In some cases, this interaction achieves timbres not possible with the equivalent duet of two physical players.

To the author's knowledge, this is the first implementation of an LSTM with an XY pad-like instrument in this manner. We hope our open-sourced code and detailed discussion of implementation in section 4 will provide a point-of-entry for members of the community interested in using deep learning in their works and creative process.

Deep learning is a new and rapidly changing field. We hope to see more applications that combine art and deep learning, collaborations between machine learning and artistic communities, as well as hardware implementations that embed these models into familiar form-factors so players can interact with these systems on tactile, physical hardware.

7. REFERENCES

[1] Elgammal, A., Liu, B., Elhoseiny, M., & Mazzone, M. (2017). CAN: Creative Adversarial Networks, Generating" Art" by Learning About Styles and Deviating from Style Norms. *arXiv preprint arXiv:1706.07068*.

[2] Jin, Y., Zhang, J., Li, M., Tian, Y., Zhu, H., & Fang, Z. (2017). Towards the Automatic Anime Characters Creation with Generative Adversarial Networks. *arXiv preprint arXiv:1708.05509*.

[3] Ha, D. “Recurrent Net Dreams Up Fake Chinese Characters in Vector Format with Tensorflow http://blog.otoro.net/2015/12/28/recurrent-net-dreams-up-fake-chinese-characters-in-vector-format-with-tensorflow/. Published 28-12-2015. Accessed 12/6/2016

[4] Eck, D., & Schmidhuber, J. (2002). Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on* (pp. 747-756). IEEE.

[5] Bretan, Mason, Gil Weinberg, and Larry Heck. "iterA Unit Selection Methodology for Music Generation Using Deep Neural Networks." *arXiv preprint arXiv:1612.03789* (2016).

[6] Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., & Norouzi, M. (2017). Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. *arXiv preprint arXiv:1704.01279*.

[7] Fiebrink, R., Trueman, D., & Cook, P. R. (2009, June). A Meta-Instrument for Interactive, On-the-Fly Machine Learning. In *NIME* (pp. 280-285).

[8] Engel, Jesse, et al. "Neural audio synthesis of musical notes with wavenet autoencoders." *arXiv preprint arXiv:1704.01279*(2017).

[9] Bretan, P. M. (2017). *Towards An Embodied Musical Mind: Generative Algorithms for Robotic Musicians* (Doctoral dissertation, Georgia Institute of Technology).

[10] Oord, A. V. D., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

[11] Pachet, Francois. "The continuator: Musical interaction with style." *Journal of New Music Research* 32.3 (2003): 333-341.

[12] Bretan, M., Oore, S., Engel, J., Eck, D., & Heck, L. (2017). Deep Music: Towards Musical Dialogue. In *AAAI* (pp. 5081-5082).

[13] Kaoss Pad Quad. Product Website. http://www.korg.com/us/products/dj/kaoss_pad_quad/ Accessed 2/1/2018.

[14] Yoo, M. J., Beak, J. W., & Lee, I. K. (2011). Creating Musical Expression using Kinect. In *NIME* (pp. 324-325).

[15] Hantrakul, Lamtharn, and Konrad Kaczmarek. "Implementations of the Leap Motion device in sound synthesis and interactive live performance." *Proceedings of the 2014 International Workshop on Movement and Computing*. ACM, 2014.

[16] Voutsinas, J. (2017). The mi. mu Gloves: Finding Agency in Electronic Music Performance through Ancillary Gestural Semiotics.

[17] Ha, David, and Douglas Eck. "A Neural Representation of Sketch Drawings." *arXiv preprint arXiv:1704.03477* (2017).

[18] Roli Lightpad Block. Product Page. https://roli.com/products/blocks/lightpad-block . Accessed 10-1-2018

[19] Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM.

[20] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[21] Xingjian, S. H. I., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems* (pp. 802-810).

[22] Zaytar, M. A., & El Amrani, C. E. (2016). Sequence to sequence weather forecasting with long short term memory recurrent neural networks. *Int J Comput Appl*, *143*(11).

[23] GestureRNN Demo Video link: https://www.youtube.com/watch?v=VgoVGpllaSY