10701 Project - Rock Music Generation

Adam Smoulder, Binghao Deng, Wanhe Zhao
Carnegie Mellon University
Pittsburgh, PA
{asmoulde, binghaod, wanhez}@andrew.cmu.edu

Abstract

While the challenge of automated music generation has been tackled by many groups previously, few have focused on the task of creating rock music using machine learning algorithms. In this work, we used a long short-term memory model (LSTM) to first learn the structure of individual rock music instrument tracks, and we generated basis tracks from these. Songs were made by putting this basis track through recurrent neural networks (RNNs) trained to "classify" a given instrument with labels for others (e.g. a generated bass track would yield a classified guitar track), leading to cohesive instrumentation. Preliminary surveys ranked our generated tracks with quality in between that of real rock songs and tracks created with a random basis track. While our result may not have produced pieces that are sonically indistinguishable from human productions, it is an early step in the direction of automated rock music generation.

1 Introduction

Throughout the animal kingdom, songs have been used for eons to accomplish everything from courtship to marking one's territory [1,2]. It is humans alone, however, that not only have a sense of mathematical cognizance about the music we create, but also utilize instruments external from our body in groups to compose synergistic songs. While various societies have developed complex analyses of musical understanding and compositions associated with these, only recently has the possibility of "intelligent" automated music development been in grasp. There are a variety of applications for this automatic type of music generation, such as giving musicians a starting point for making a new song, filling in an instrument for an incomplete piece, or even giving sporting events a mechanism to create royalty free tracks. Common musical structures among many modern genres allow for musical features to be characterized and labeled for songs (e.g. consistent key signature, common chord progressions) [3], while modern computing and machine learning methods provide an avenue for identifying these features.

The first piece of music automatically generated by computers in Bell Laboratories in 1957 using Markov chains does not quite resemble anything that most would consider "music" in a modern sense [4, purportedly at `https://www.youtube.com/watch?v=PM64-lqYyZ8`]. More recent developments in machine learning methods and computing provide valuable tools to possibly achieve music learning and generation of more pleasing songs. Of particular popularity have been recurrent neural networks (RNNs) due to their capacity for incorporating history in time series input information to a neural network learning framework [5,6], along with a modified version of RNNs called long short-term memory networks (LSTMs) that better permit learning of long term dependencies [7,8]. LSTMs in particular have been utilized by a variety of groups, most often for learning and generation of piano pieces (see Background and Related Works section). More broadly, these applications have tended to be designed for individual instrument track creation based on a variety of learned genres.

One genre of music that has received little attention (to our knowledge) in automated music composition is rock and roll, or "rock" for short. Featuring a broad scope of melody, emotion, and primal

Submitted as Introduction to Machine Learning (10-701, Fall 2018) Final Project Report. Do not distribute.

awareness, rock music has had a defining role in recent cultural history and maintains a place in the hearts of many today [9]. Often, these compositions host only a few instruments, such as an electric guitar, a bass guitar, and a drum kit. But what makes rock music great? Are there dimensions that can be isolated from its works that could be distilled and then recombined to yield novel compositions of the same emotional salience, like adding vectors of a basis set? While this may be a stretch, it is not unreasonable to consider that certain structures native to rock music could be learned with machine learning algorithms and then applied to generate new songs in an automatic matter.

In this work, we used a large corpus of rock tracks to learn the structure of single-instrument tracks and then generate polyphonic (bass and guitar) pieces accordingly. We first trained LSTMs to generate both individual note and chord sequences based off of training with rock bass and guitar tracks. We then tried to mirror an approach often used in real rock music composition, where one member of the band writes an initial instrument part (what we call "basis tracks") and the other band members build their instrument tracks off of that (our "classified tracks"). LSTM-generated basis tracks were fed into RNNs trained with the basis instrument as input and the opposite instrument as output, allowing creation of new instrument tracks by "classifying" the basis tracks. The combined compositions were presented in a blinded survey to a variety of peers to evaluate how realistic generated songs were compared to those made from random noise basis tracks and to real songs' bass and guitar parts. Our goal was to produce bass and guitar tracks that were cohesive in their presentation rather than disparate and individualized.

2 Background and Related Work

Our initial inspiration for the project came from the "Hello World" Album [10], which used Artificial Intelligence for composing music. The team first released the song, Daddy's Car (`https://www.youtube.com/watch?v=LSHZ_b05W7o`) in the style of The Beatles. Even though the approach used was rather supervised, being the result of the cooperation with artists and singers, this album is an impressive work showcasing some high-quality possibilities of semi-automated music generation. This is only one example of a litany of works that have been performed using machine learning techniques to generate music (see [4]).

In terms of using MIDI data for music learning, a recent blog post from Halil Erdogan gave us great insights into data preprocessing using existing Python libraries [11]. While this post was focused on methods for data preprocessing on piano MIDI files, we were able to build on this to preprocess our rock MIDI files. We also found great value in Christine McLeavy's blog post on chord-wise and note-wise representation in the choices of encoding MIDI files [12].

For model selection, two recent blog posts from Andrej Karpathy and Sigurður Skúli [13,14], have shown the promising results using LSTM models and motivated our main direction on algorithm selection with their clear explanations of their implementation. Further, we tried to incorporate some language model-associated work, such as that by Merity, Keskar, and Socher which proposes the weight-dropped LSTM as a form of regularization and optimization to help prevent overfitting across the recurrent connections [15]. The strategies proposed in this paper are compatible with other sequence learning tasks, including those which we utilize. Magenta, which spun out of Google DeepMind, developed a Drum RNN using an approach somewhat akin to this by applying language modeling to drum track generation using an LSTM [16]. This model represents polyphonic drum track as single note sequences.

3 Methods

3.1 Training Data Acquisition and Processing

For input song data format, we decided to used Musical Instrument Digital Interface (MIDI) files for their compact representation of relevant musical characteristics, such as pitch, note duration, note offset, and instrumentation. Complex sonic factors, like harmonic ratios and audio waveform shapes, do not need to be learned in this format such that models can focus on learning the aforementioned relevant factors. Most rock and punk music is polyphonic (multiple notes at a time) with electric guitar and bass being the typical pitched instruments present in compositions. Therefore, to allow splitting and labeling of each instrument track with multiple notes played at once, MIDI is much more feasible to manipulate and implement than raw audio file formats, such as .mp3.

For training and testing datasets, we acquired over 1000 MIDI songs from midiworld.com by bands typically classified as rock, punk and metal (see Supplemental Files > trainingMIDIFiles). We used the music21 Python library to parse a given MIDI file and decompose it into its single instrument tracks [17]. As seen in Figure 1, each instrument track is mapped and truncated to a 128 (pitch) by 1500 (time) matrix, where each timestep is a sixteenth note. A value of 1 (black) at any of the 128 pitches indicated the beginning of the corresponding note being played, while 0.5 (grey) was used to represent the continuation of a note, and 0 (White) indicated the note not being played. We used Musescore 2.0 used for sheet music visualizations like that seen in Figure 1 [18].

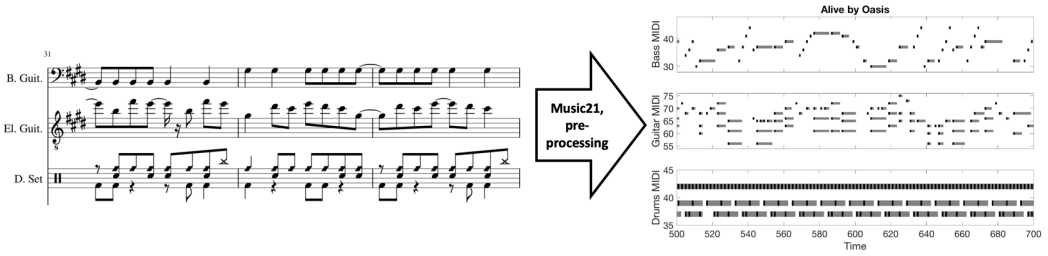

Figure 1: Example of music (MIDI) to matrix interpretation (shown for illustration purposes only). Black indicates the start of a note for the pitch determined by the y-axis location and grey corresponds to sustain. Drums were not used in the primary results of this study.

3.2 Basis Track Generation

LSTMs are widely used for modelling and generating temporal data, and accordingly, we used a simple LSTM network as the basis track generator. First, we kept music note numbers sequences and inserted a separation indicator between same notes that are played separately, such that each note can have its duration information represented by the length between different numbers. This permitted us to distinguish a note continuing from the note being repetitively played, the latter of which is common in rock music. In this, our method takes a similar approach to language learning models.

We started with a monophonic music generator for bass (Figure 2A). The input was a sequence of bass notes and the corresponding labels were the next notes in the sequence. At each timestep, the note went go through an embedding layer that embeds the note number into a vector of length 256, a 2-layer LSTM with hidden size 256 and a softmax layer to predict the probability distribution of the notes for the next timestep. To further improve our results, we also applied some regularization techniques, including variable length backpropagation and weight tying that used by Merity, Keskar, and Socher [15]. Variable length backpropagation ensures that every element will have a chance to have its error being backpropagated, while weight tying shares the weights of the embedding layer with the softmax layer to reduce the total number of parameters to be learned.

During the training process, we concatenated the songs into a single sequence and generated batches of variable lengths, which are sampled from a normal distribution. We selected the mean sequence length to be 70 with a probability of 0.95 and 70 / 2 = 35 with a probability of 1 - 0.95 = 0.05, with the variance set to 5 (see Merity, Keskar, and Socher for details) [15]. We used cross entropy loss between predicted probability distribution and the true labels, and we trained the networks using the Adam optimizer with learning rate 1e-3 and weight decay 1e-6 for all the experiments.The code is written in PyTorch and is trained on AWS p2.xlarge machine. We used a ratio of 90:10 to split the training and validation sets.

For generation, we first randomly chose from the starting notes of all the songs in our training set, and for each timestep, we sampled from the probability distribution that is generated by the network given the current input to get the next note (Figure 2B). This continued until it reached a predefined length.

After completion of a note-wise generator corresponding to bass parts, we developed a chord-wise generator for guitar pieces. We encoded the chord information by concatenating the notes of the chord into a single token and treated the unique tokens as the vocabulary for guitar music sequences. In total, we had 5463 tokens, and we tried to model it with a more complex network structure than

the individual note generator. We used a 3-layer LSTM and increased the embedding size and the hidden size of LSTM to 512. Aside from these differences, implementation proceeded identical to that of individual note generation.

We also explored generating tracks based off of solely one bands' songs. Time limitations permitted us to pursue this method with one band; we chose Green Day due to their relatively consistent and simple song structure, along with a large quantity of songs to use (>70).

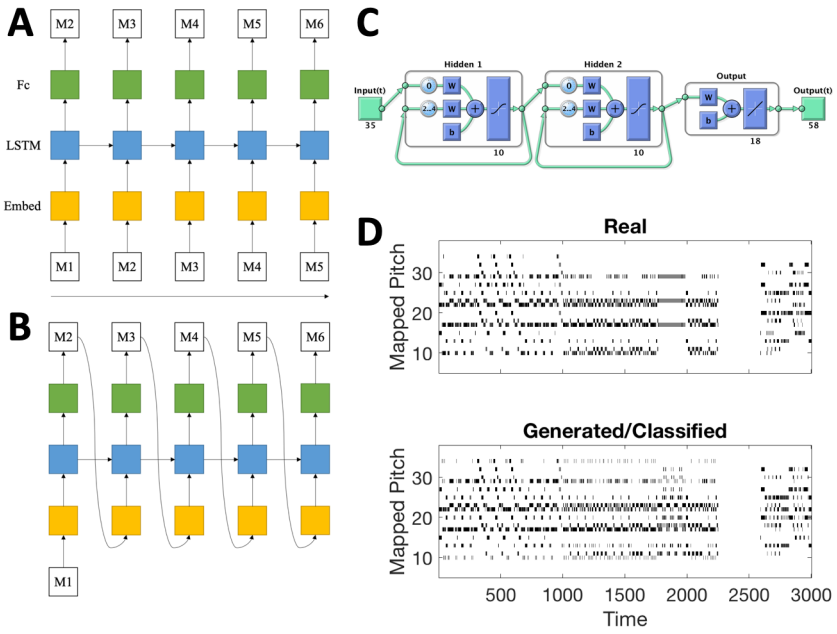

Figure 2: Models used for generation and classification. (A) LSTM network used for learning single note sequences for bass guitar parts. (B) Generation scheme using LSTM. (C) Example RNN model used for bass classification to create guitar part. (D) Visualization of 2 test set songs' actual guitar part compared to results classified from the corresponding bass tracks by the RNN.

3.3 Classified Track Creation

Once a given basis track was generated, a corresponding classified track of the opposite instrument was created using a RNN classification model implemented with the layer RNN function in MATLAB. For example, a bass basis track would be used to produce a guitar classified track. Simple RNNs were used in this case to yield more direct temporal cohesion with the given basis instrument, as opposed to LSTMs which would incorporate undesired longer term effects. Time lags of 2 and 4 timesteps were re-input to the model, as most musical structure occurs in sets of quarter and eighth notes; other attempted lag values did not affect loss appreciably in our cursory tests. Due to time constraints, we trained RNNs on strictly on one band's songs (we once again selected Green Day). A train to test ratio of 1:4 was used, as we found that only a small number of songs was needed in training to yield sufficient validation.

Both the bass-to-guitar and guitar-to-bass RNNs were only 2 hidden layers of 10 neurons each with tanh activation functions (Figure 2C). We found that increasing the number of layers and neurons not only increased computational load (as expected), but it also tended to produce classified tracks that were rather stagnant in nature. For instance, one attempt using three layers of 15 units each to classify bass inputs to a guitar part yielded constant 16th note chords on the guitar corresponding to the bass note played. This is hypothesized to be a function of overfitting to the training set, where a particular set of Green Day songs are very likely to exhibit this pattern.

To further avoid this putative overfitting, we also limited the number of iterations of the RNNs (20 iterations). The scaled conjugate gradient method was used to update weights for speed of iterations with a mean-squared error loss function. While cross entropy loss was initially attempted as a more

intuitive loss, it tended to result in a constant-value note repetition, whereas mean-squared error (MSE) produced impressively accurate results on test data near immediately (see Figure 2D for example of MSE-loss results). Due to time constraints, we stuck to this framework to continue. Once the basis track and classified track were created, they were combined in Musescore and exported as a .mp3 file for future use and evaluation.

4 Experiments and Results

4.1 Model Performance

We used two datasets for training the basis track generators. The first one is a full collection of rock music songs, while the other is the subset written by one band (Green Day, 88 songs). In total we have extracted 863 tracks for bass pieces and 938 tracks for guitar pieces. For the bass generation model, we applied dropout rate of 0.65 in the two layer LSTM and trained it for 10 epochs. Testing and training loss both dropped below 1 for both the full dataset and the Green Day only trained models (Figure 3A for Green Day, see Supplemental Files for full set losses).

For the guitar generator, the losses were higher compare to the bass pieces, so we lowered the dropout rate to 0.5 expecting to have a better result. Another change is that instead of splitting the training and validation sets by songs, we splitted the sequences for each song based on the ratio and combined them together. This mostly likely because different songs are using different sets of chords and the original split mechanism will result in a large gap between training and validation results. We trained the network on the full dataset for 20 epochs. Guitar generation did not achieve the same quantitative loss as bass note generation, as even though training loss went below 1, validation loss converged to 1.2. For Green Day only experiment, there are 89 songs, 284 unique combinations of chords. This reduced vocabulary allowed for the training and validation losses to go below 1 for the guitar generation model (Figure 3B, see Supplemental Files for full set loss).

We found that the model with a loss below 1 was able to generate meaningful pieces during the generation process qualitatively. Otherwise, the model tended generate a long sequence of repeating notes (especially zeros) without any reasonable variations.

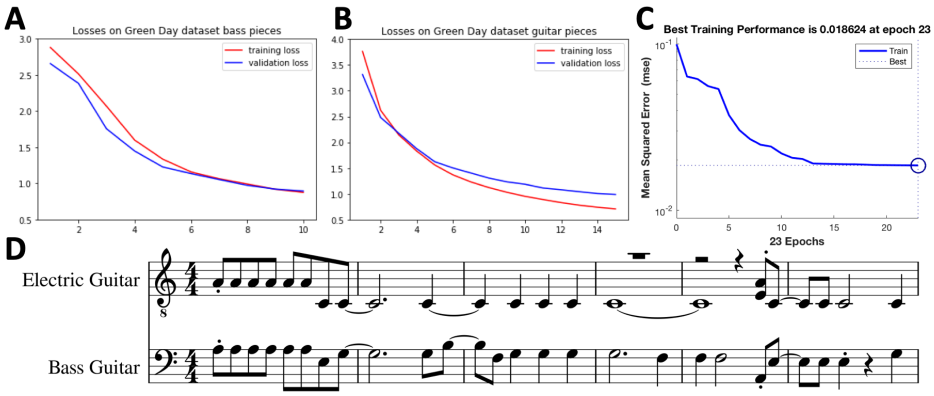

Figure 3: Algorithm performance. (A) Cross-entropy loss of LSTM model for bass track generation with one band, and (B) for guitar track generation. (C) MSE loss for bass-to-guitar RNN model. (D) Example of combined output track from the two models.

RNN training error against iterations for the bass-to-guitar classifier converged to a MSE value of 0.0186 (Figure 3C). Corresponding test error was 0.0193. Similar, though slightly lower, values were observed for guitar-to-bass classification (0.0132 and 0.0153 for train and test error, respectively). After approximately 20 iterations, changes appear to be rather minimal, though models were stopped at this point to cull any further overfitting. While this quantitatively seemed satisfactory, we also examined matrix representations of classified tracks similar to that in Figure 1 alongside basis tracks to assert no odd local optima solutions were adopted (i.e. all 0s, all 1s).

4.2 Song Quality Evaluation

To evaluate the quality of our model, we decided to use similar approach as used in [19]. We conducted a blinded survey, and asked 15 volunteers to rate 10-30 seconds clips of MIDI songs with guitar and bass tracks on a scale from 1 (random noise) to 10 (professionally composed). See Supplemental Files > Survey Songs for the audio files from the survey. 12 samples are given to the users, containing a combined tracks of MIDI bass and guitar compositions, and are distributed as follows:

- Four samples from real rock songs (Real songs)
- Two samples from songs based on a random basis track (Noise basis songs)
- Eight samples from songs with basis tracks generated by the LSTM and classified tracks created with the RNNs (Generated songs)

For creating the noise basis track, we used prior probability from training data for whether or not a given time step should have a note played, and if yes, we made a random draw from histogram of notes. Repeating this for all timesteps generated a sequence of notes, which we passed into the appropriate RNN classifier to generate a noise song.

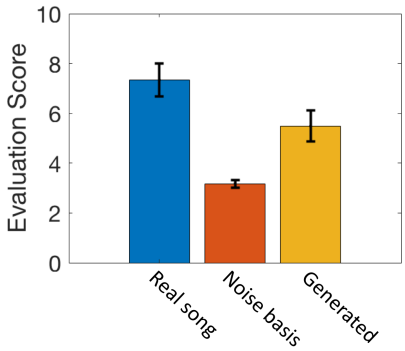

Figure 4: Average survey evaluation scores for 15 participants (error bar is standard deviation). No significance was determined due to small sample size.

We averaged the evaluation scores within each type of sample songs, the results of which are shown in Figure 4. Generated tracks scored 5.49+/-0.63 (mean+/-std), which is in the middle of noise basis tracks (3.18+/-0.15) and real tracks (7.34+/-0.67). One interesting note was that the best generated song score (6.57) was higher than lowest real song score (6.36). The results overall indicate that our model could generate tracks that are at minimum distinguishable from random noise, though certainly not always as realistic as true songs.

One incoming expectation was that tracks formed from a guitar basis track would do better than a bass basis track, since creating cohesive single note classified tracks from a chord basis would be intuitively easier than the opposite direction. However, this hypothesis on "information loss" from comparing guitar basis to bass basis is not shown in the participant responses, where songs generated by the two approaches scored approximately equal. This could be due to using a band with limited repertoire for classified tracks such that chords were easy to predict from a bassline.

5 Conclusions

There are a fair number of limitations to be addressed for the presented work in both its implementation and interpretation. In data, the MIDI file format was chosen for its ease of use in learning processes; audio quality, however, is not up to par with actual standard recorded audio waveforms. The majority of our other limitations pertained to meeting time constraints. For example, we only explored rather vanilla LSTM models without customized functionality or other features. In the same vein, limitations were made on the RNN models in a rather ad-hoc matter that yielded qualitatively appropriate results as opposed to scientifically trying to find the "correct" model parameters. Our survey was also very coarsely designed to try to encourage ease of participant response such that we had results to report

in our time frame. This motivated the decision of 10-30s clips and only 12 tracks total, an admittedly rather limited sample set. We also only obtained responses from 15 participants from an uncontrolled population. There may be a variety of latent biases that we are unable to address from this setup. Finally, while we attempted to get fair ratings by blinding the categories in our survey, there was technically no baseline track to compare to for participants. Even with the given technical and survey limitations, the reader can decide for themselves if the songs produced through our algorithms are not of the same quality as the randomly generated tracks (see Supplemental Files > Survey Songs).

While the present work produced rock music pieces that were moderately successful in survey, there are a great deal of other directions that stem from this point which could lead to great improvement. One important topic not examined in this paper was the generation of appropriate drum tracks. We had some degree of success in this regard (see GD_BGD_gen1.mp3 in Supplemental Files for an example) using RNN classification over short blocks of four timesteps as inputoutput, though were unable to generate diverse enough patterns or overcome bugs within the given time limitations to yield markable results. We also considered the value of differing loss functions based on musical prior knowledge (e.g. wrongly classified notes in the correct key signature would receive less penalty) and feeding back classified tracks into opposing RNNs to establish a "feedback loop" akin to two musicians building off of each others' ideas. Long-term song structure, such as incorporation of verse and chorus sections, would be another interesting topic to explore, possibly using some flavor of state-based learning model. These all are fascinating avenues for future groups to explore.

We have created a rock music generator by first using LSTM models to learn bass and guitar track structure from MIDI files, then generating an individual track from this knowledge, and finally building upon it with RNNs trained on bass and guitar data. Though it does not produce music equal in quality to that of real bands (yet!), it provides a step in the direction of automated music generation of one of the most beloved genres to mankind.

Supplemental Files

Available at `https://cmu.box.com/s/6opsbw2ug1gbko7xkauadzzz7peq4gvm`.

References

[1] Rothenberg, D. "Whale music: Anatomy of an interspecies duet." *Leonardo Music Journal.* **18,** 47–53 (2008). Available at: `https://www.sibetrans.com/trans/articulo/97/to-wail-with-a-whale-anatomy-of-an-interspecies-duet`.

[2] Rothenberg, D., Roeske, T. C., Voss, H. U., Naguib, M. & Tchernichovski, O. "Investigation of musicality in birdsong." *Hearing Research.* **308,** 71–83 (2014). Available at: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3947120/`.

[3] Jones, M. R. "Dynamic pattern structure in music: Recent theory and research." *Perception & Psychophysics.* **41,** 621–634 (1987). Available at: `https://link.springer.com/article/10.3758/BF03210494`.

[4] Briot, J. P., Hadjeres, G., & Pachet, F., "Deep learning techniques for music generation – A Survey." *arXiv* (5 Sep. 2017). Available: `https://arxiv.org/pdf/1709.01620.pdf`.

[5] Simon, I., & Oore, S. "Performance RNN: Generating music with expressive timing and dynamics." (29 Jun. 2017). Available at: `https://magenta.tensorflow.org/performance-rnn`.

[6] Boulanger-Lewandowski, N., Bengio, Y., & Vincent, P. "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription." *Proceedings of the 29th International Conference on Machine Learning.* 1156-1166 (2012). Available at `https://arxiv.org/abs/1206.6392`.

[7] Eck, D., & Schmidhuber, J. "A first look at music composition using LSTM recurrent neural networks." Technical report, IDSIA/USI-SUPSI, Manno, Switzerland (2002). No. IDSIA-07-02. Available at `http://people.idsia.ch/~juergen/blues/IDSIA-07-02.pdf`.

[8] Hochreiter, S., & Schmidhuber, J. "Long short-term memory." *Neural Computation,* **9(8),** 1735-1780 (1997). Available at `https://www.bioinf.jku.at/publications/older/2604.pdf`.

[9] Wikipedia contributors, "Rock music." *Wikipedia.* (2018). Available at: https://en.wikipedia.org/w/index.php?title=Rock_music&oldid=873391582.

[10] Flow Records, "Hello World." Digital Album. Available at: https://www.helloworldalbum.net/.

[11] Erdogan, H., "My journey to music-speech with deep learning (part 1) - Music generation with LSTM." *Hedonistrh's Space (Blog).* (2018). Available at: https://hedonistrh.github.io/2018-04-27-Music-Generation-with-LSTM/.

[12] McLeavy, C., "CLARA: A neural net music generator." *Deep Learning Explorations (Blog).* (2018). Available at: http://christinemcleavey.com/clara-a-neural-net-music-generator.

[13] Karpathy, A., "The unreasonable effectiveness of recurrent neural networks." *Andrej Karpathy Blog.* (2015). Available at: http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[14] Skúli, S., "How to generate music using a LSTM neural network in Keras," *Towards Data Science (Blog).* (Dec. 2017). Available at: https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5.

[15] Merity, S., Keskar, N. S., & Socher, R. "Regularizing and optimizing LSTM language models." *arXiv.* (2018). Available at: https://arxiv.org/abs/1708.02182.

[16] "Magenta," *Magenta.* Available at: https://magenta.tensorflow.org/.

[17] M. S. Cuthbert, "music21 documentation," *music21 - music21 Documentation,* 29 Oct. 2018. Available at: http://web.mit.edu/music21/doc/index.html.

[18] "Musescore 2.0." *Musescore.* Available at: https://musescore.org/en.

[19] Huang, A., & Wu, R., "Deep learning for music," *arXiv.* (2016). Available at: https://cs224d.stanford.edu/reports/allenh.pdf..