**Introduction:**

Our goal behind this particular project would be to use a generative model neural network in order to generate new, original music. <br/>\n",

"This music would not be a full length piece, and would likely only last for four measures or so. <br/>\n",

"Currently we imagine using the music from the popular NES game Castlevania as our base dataset, however I predict this would be easy to expand if we chose to do so in the future. <br/>\n",

"Needless to say, attempting to engage something constructed from pure logic and push it towards creative pursuits is fascinating, and I am eager to see how far we can take it."

**Methods:**

*Dataset:*

Thousands of musical instrument digital interface (MIDI) formatted files from video games produced by the consumer electronics company Nintendo are available online through VGMusic [1]. Of these files, 177 files from the Nintendo Entertainment System (NES) game *Castlevania* may be used as the data set. Notably, there are repeated songs in this data set which will have to be removed, so it is very likely that MIDI files from other NES games will have to be sampled. However, the sizes of data sets in other studies varies widely, but at least 24 files (either from *Castlevania* or another suitable source) will be used in our study [2], [3], [4].

By default, files of MIDI format cannot be used to train neural networks [5], [4]. Several methods are available for encoding MIDI files; however, the *pretty\_midi* library for Python will be used for this study [5], [6]. Only melodic instruments will be sampled from a given MIDI file since MIDIs will be encoded into 2D arrays called piano rolls that, by default, ignore drum instruments on MIDI channel 10 [5].

Each MIDI file will be augmented by constraining the set notes to a single domain. More colloquially, MIDI files will all be transposed to the key of C. If it is found that this is not ideal for our study, a data augmentation protocol that essentially balances the proportion of songs in different keys will be followed as in [4]. Only MIDI files in the most common time signature (4/4) will be included as in [5].

To judge the performance of the network, 70% of the data set will be used for training and 30% will be for testing.

*Model:*

Recurrent neural networks (RNNs) are unlike feedforward neural networks in that RNNs possess connections pointing both forward *and* backward in the network [7]. Vanilla RNNs notoriously suffer from the vanishing gradient problem, which prevents weight-updates in earlier layers of multilayer networks. The resolution of this problem via *long short-term memory* (LSTM)architecture has since made LSTM-RNNs ideal for many tasks including music generation [5], [8]. Since musical melodies are inherently contextual—that is they rely on the context of previous notes to produce a coherent melody—some recursive update of weights is a necessity. LSTM-RNN extends the memory of a traditional RNNs by using memory cells that are capable of “remembering” information in more than just the immediately previous layer [4].

The LSTM-RNN will initially be constructed following several architectural considerations from [9]. The *Keras* API (*TensorFlow* backend) will be used for the implementation of the network [10]. The neural network architecture will consist of an input layer of size determined by piano roll formatted data, a single LSTM layer consisting initially of 256 hidden units, and an output layer with softmax activation function [5], [11]. The model will be compiled using categorical cross entropy as the loss function and RMSProp (or possibly Adam) as the optimizer [4], [12]. The output of this network will be 4 – 8 measures of music as in [5].

*Performance metrics:*

The generation of music is a multi-class classification problem (that is distinguishing from more than just two classes, for example, as in binary classification) where each class is a unique pitch and time from a MIDI piano roll matrix [5], [12]. Therefore, classification metrics such as the *F-measure* are appropriate for assessing the accuracy of the model [4], [13]. In this way, remixes of Castlevania songs (available in data set) may be used as targets to quantitatively assess the similarity of generated songs.

Several other quantitative metrics based on musical domain knowledge may also be included such as those comparing tonal tension and interval frequencies, the statistical analysis of labeled musical events, and/or those involving tonal distance, rhythmic patterns, and pitch classes [14].

Qualitative metrics may be implemented via a survey where each member of the CSCI 4850 class is asked to evaluate 3 samples of music. Each member will be told that at least one of the samples is from a *Castlevania* game and at least one of the samples is generated by the model. Each member will be asked to evaluate the quality of the sample from 1 to 10 where 1 indicates poor quality and 10 indicates very good quality. This is akin to a “musical Turing test” devised by [5] and subject to modification (e.g., participants asked to select which model they think is generated and the degree of confidence (1-5) they have in their decision—surveyors may be given all placebo samples in this design).

*Considerations*

# It may be significantly more feasible to train a model using data that is from a single polyphonic instrument (such as a piano). Designing the LSTM-RNN is more intuitive as it predicts the next sequence of notes from a constrained set of say 88 (in the piano case) notes [15]. Alternatively, only a single MIDI channel could be selected to constrain the model to a monophonic instrument. Qualitative testing may also occur intermittently within the team to assess the \*musical quality\* of generated samples.

# References

|  |  |
| --- | --- |
| [1] | M. Newman, "Video Game Music Archive: Nintendo Music," VGMusic, 1996. [Online]. Available: https://www.vgmusic.com/music/console/nintendo/nes/. [Accessed 08 March 2021]. |
| [2] | A. Huang and R. Wu, "Deep Learning for Music," Stanford University, Stanford. |
| [3] | S. Skuli, "How to Generate Music using a LSTM Neural Network in Keras," Medium, 7 December 2017. [Online]. Available: https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5. [Accessed 8 March 2021]. |
| [4] | S. AlSaigal, S. Aljanhi and N. Hewahi, "Generation of music pieces using machine learning: long short-term memory neural networks approach," *Arab Journal of Basic and Applied Sciences,* vol. 26, no. 1, pp. 397-413, 2019. |
| [5] | N. Mauthes, *VGM-RNN: Recurrent Neural Networks for Video Game Music Generation Generation,* Master's Projects, 2018, p. 595. |
| [6] | C. Raffel and D. Ellis, "Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty\_midi," in *15th International Conference on Music Information Retrieval Late Breaking and Demo Papers*, Taipei, 2014. |
| [7] | A. Geron, "Chapter 15: Processing Sequences Using RNNs and CNNs," in *Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed., Sebastopol, O'Reilly Media, Inc., 2019, pp. 497-499. |
| [8] | S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 6, no. 2, pp. 107-116, 1998. |
| [9] | S. Mangal, R. Modak and P. Joshi, "LSTM Music Based Generation System," *International Advanced Research Journal in Science, Engineering and Technology,* vol. 6, no. 5, pp. 47-54, 2019. |
| [10] | P. W. D. Charles, *Keras,* Github, 2013. |
| [11] | A. Ycart and E. Benetos, "A Study on LSTM Networks for Polyphonic Music Sequence Modelling," in *18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, 2017. |
| [12] | R. Vidiyala, "Music Generation Through Deep Neural Networks," Medium, 21 October 2020. [Online]. Available: https://towardsdatascience.com/music-generation-through-deep-neural-networks-21d7bd81496e. [Accessed 8 March 2021]. |
| [13] | A. Geron, "Chapter 3: Classification," in *Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed., Sebastopol, O'Reilly Media, Inc, 2019, pp. 100-106. |
| [14] | L. Yang and A. Lerch, "On the evaluation of generative models in music," Georgia Institute of Technology, Atlanta, 2018. |
| [15] | D. Kang, J. Kim and S. Ringdahl, "Project milestone: Generating music with Machine Learning," Stanford University, Stanford, 2018. |