Music Generation with Recurrent Neural Networks

Malakhi Hopkins, Ryan French, Kavish Shah

Abstract—**Deep learning focuses on classification and generation of various types of data. Computationally music can be stored in the form of MIDI files and be easily manipulated with production software. In this paper we aim to use Deep Learning to generate music based on existing MIDI data that can then be used as a baseline or melody within more complex production software. To accomplish this we train both a Recurrent Neural Network (RNN) which utilizes the temporal nature of music and a Variational Autoencoder (VAE) which generates original data from trained distributions on pianoroll MIDI data in Python using the open source framework Pytorch. Using these Deep Learning Methods, we were able to generate MIDI files that qualitatively resembled our data.**

I. Introduction

Deep learning has aided in the classification and generation of a wide variety of datasets. Images, speech, video, traffic, sports, healthcare, and other fields can all be quantified and passed as valid data into deep learning models. Music generation follows this same trend but is a relatively underdiagnosed field, especially when considering the quantitative nature of music.

MIDI files, at the core of the digital music industry, encode what used to only be qualitative inputs as arrays of quantitative data. Prior to the digital music revolution, it would be near impossible to train a network on data from an analog waveform replicator such as a vinyl record or cassette tape. With the advent of CDs, and later mp3 and MIDI files, the digital representation of waveforms allow data scientists and music theorists to take a much more analytic approach to decomposing music.

Music's temporal nature makes it a prime candidate to adopt models similar to ones used to study natural language processing (NLP). Of the most popular NLP models are Recurrent Neural Networks (RNN), which take in a sequence of data and predict a new sequence, comparing the two with a loss function. Another popular and quite recent development in deep learning are generative models such as Variational Autoencoders (VAE), which seek to generate original data points from trained distributions.

II. Prior Work

Several notable independent researchers have published their music-generating deep learning networks, leading to a broad array of network components from which to choose. A large portion of effort needed to train such networks for music generation is deciding how to construct a dataset. It is imperative to control for the following variables found within all midi files: for each note played: Is this a note or a chord? How long should this note be played? On which octave is this note played? One variable can be accounted for by establishing a constant timestep between notes, and a second variable by encoding each note/chord as scalar number as opposed to representing octaves as its own dimension [1].

A simple melody of piano notes can be overlayed with a constant drum or rhythm sequence to create an attractive sound, from which we drew much of our inspiration [2]. Convolutional architectures have also proven to be effective; instead of focusing on the temporal nature of RNNs, one can focus on transforming a given sequence to an attractive output, one measure at a time [3]. Generative Adversarial Networks (GAN) have also been created with success; they add a discriminative neural network on top of a generative network found in VAEs [4].

Two UPenn undergraduates created a collection of networks, including all listed above (RNN, CNN, VAE, and GAN) off a popular open source music dataset, the Lakh Pianoroll Dataset (Lakh means one hundred thousand in the Indian numbering system). They cleverly approached music generation via several avenues, all using a five instrument model. This adds an additional dimension of complexity in how an orchestra is assembled: Should one instrument act as the lead predictor? Are all five instruments composed from a central directive engine? They answered both questions in an elegant, quantitative manner[5].

III. Dataset

The Lakh Pianoroll Dataset (LPD) is a collection of 174,154 multitrack pianorolls derived from the Lakh MIDI Dataset [6]. Compiled by Academia Sinica, a research firm in Taiwan, LPD consists of 5 batches of pianorolls, some being thoroughly labeled and some as raw data. We used the LPD-cleansed dataset, which is a subset of the full LPD library and contains 21,425 pianorolls. Every pianoroll in LPD-cleansed is paired with a dictionary of song attributes that can be downloaded from the Million Song Dataset. [7].

From our subset of 21,425 pianorolls, we took only samples whose instrument was labeled 'Piano'. We ended up with 1189 pianorolls, each ranging in length from 137 to 2393 notes, for a total of 981,961 notes. Digital music is most commonly encoded by assigning each pitch, or note, an integer value between 0 and 182, inclusive. Our pianoroll data is cleaned to encode such data as near multi-hot vectors, with each datum ranging from 0-127, representing the note's velocity (volume). To eliminate this velocity, simplifying our model, we normalized each note on a linear scale of [0,1]. We therefore end up with a large multi-hot tensor of dimensions (981961, 182) from which we can split and sample.

The simplicity of our cleaned, encoded, and labeled dataset offers the perfect opportunity to train networks used similarly for Computer Vision or Natural Language Processing. Now that our audio information is represented by tensors, we can implement popular PyTorch libraries to better understand how melodies can be learned.

IV. Methodology

Our first approach to Music Generation consists of a Recurrent Neural Network (RNN), which is trained on a sequence of notes within a MIDI file and predicts a given number of future notes based on the input sequence. This RNN is an updated implementation of the RNN we created in our Homework 3, an RNN with one hidden layer which predicts the one-hot vector of the next note(s) as output, given the past sequence. Initially, we recognized our dataset as a large series of 182-long tensors. However, we discovered that structure of each tensor represented a moment in time and contained multiple notes and their velocities; each was not actually a one-hot tensor. This set of multiple notes, or chords, proved to be too large for reasonable one-hot conversion, so in order to simplify the notes, we decided to take only the lowest note, the root of the melody, that was being played at each moment of time. This resulted in a one-hot vector of the 128 MIDI notes. Next we trained our model on various sequence lengths between 2-24 and with various learning rates between 0.001 - 0.01 to identify the most desirable performance of the model. Our final model ran with a sequence length of 10 and a learning rate of 0.01. As expected, we ran into difficulty with the model producing the same repeated notes, however, we decided to increase the future prediction length and this lead to more variation within our notes.

We also tried to model our MIDI data using Variational Autoencoder(VAE). Our idea was to obtain a latent space for our notes and then experiment on it by picking random samples from the latent space and passing them through our decoder and then joining them together. The temporal aspect of the music was not captured, but there was a chance that the randomly picked samples when decoded and played together might give rise to unconventional but pleasant music. We started by arranging the notes from all the tracks in our data into a single tensor. The encoder and decoder each consist of a single hidden layer. We tried three different approaches for designing the loss function of our encoder. Each note is 128 dimensional and each entry in the note can take a velocity value between 0 and 127, the same structure as for our RNN. We treated different velocity values as classes and designed a loss function combining the cross-entropy loss of reconstruction and KL divergence loss. Next, we tried to assemble an MSE loss and KL divergence loss. Finally we took inspiration from the method proposed in HW5 and binarized our notes and use Binary cross entropy loss along with KL divergence loss. We found that the last method gave us less noise while and better result once we convert the output to MIDI files. We used the latent space prior to generate random samples. We passed these samples through the decoder to test the music generated by our model.

V. Results

We generated a few MIDI files as music samples and they can be found here [8]. Our final result for our RNN is sequence of notes that eventually leads to a distinct repeating melody that persists until the end of the song. This structure seems to resemble that of a song with a distinct introduction and then a repeating baseline/melody that follows afterward, which is very common in music. The result for our VAE is a collection of generated notes that tend to sound pleasant but can sometimes be incoherent.

The following are plots that we generated from training with different learning rates and from reconstructing binarized notes:

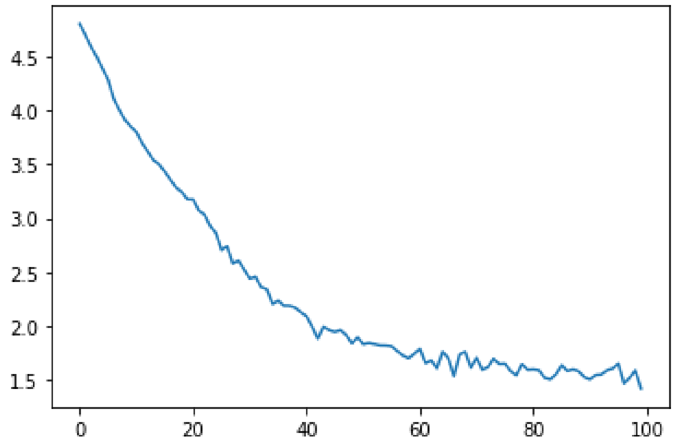

Fig. 1: NLL Loss vs Percentage of Iterations Completed, Learning Rate = 0.01

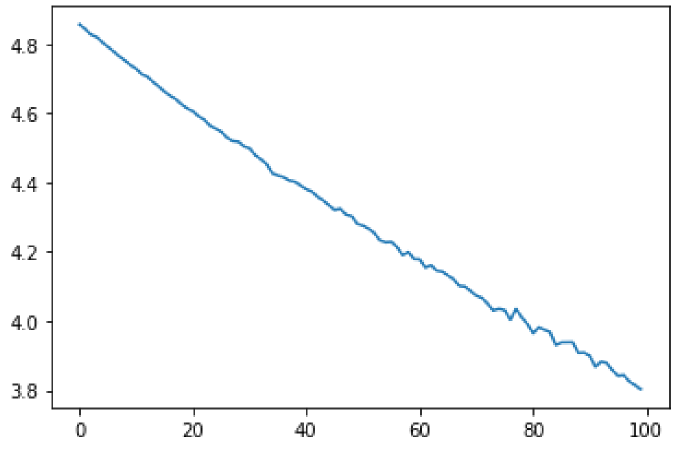

Fig. 2: NLL Loss vs Percentage of Iterations Completed, Leaning Rate = 0.001

VI. Conclusion and Future work

In this paper we used Deep Learning to generate music based on existing MIDI data that can then be used as a baseline or melody within more complex production software. We accomplished this through both a Recurrent Neural Network (RNN) and a Variational Autoencoder (VAE) within Python using the open source framework Pytorch, and were able

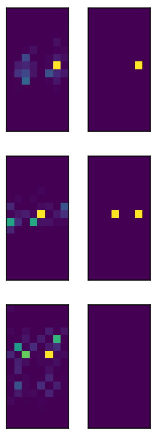
Fig. 3: Reconstructed(left) vs Original images of binarized(right) notes

to generate MIDI files that qualitatively resembled our data, which we have provided links to in our results section.

Future work that could build upon our results is creating another RNN that trains on the chords for each note throughout the training data and once given an sequence of notes, provides the expected supplementary chord notes for each baseline note from the original RNN. A drum line could also be predicted through another RNN the takes in the baseline as input. Alternate Models could be used to predict future notes, such as GAN, CNN, etc., which could then be compared with one another.

The VAE model has no way to capture the temporal nature of music and thus we observe that the notes generated, though pleasant individually, are sometimes incoherent. We tried to think of ways in which we can incorporate the temporal nature in our VAE, and wanted to integrate an RNN within our VAE. The idea was to first completely train our VAE and define a latent space prior, then generate the latent space representation of our complete dataset. Once we have the latent space representation we can use chunks of it to train the RNN. Unfortunately, we could not implement it in time for our project but we believe it would be the next logical step in scope of work.

Further, We were not able to incorporate other instruments, nor the velocity of the piano notes in our current work. One of our future steps would be to include those effects in our model to improve the quality of our music.

VII. Acknowledgements

A special thank you to Professor Pratik Chaudri and all Teaching Associates of UPenn ESE 546 for providing us with the foundational knowledge of deep networks and the resources to train our own.

Thank you as well to Isaac Pham and Matthew Kim, both UPenn undergraduate ’22, who published an outstanding report on medium.com and offered personal guidance as we embarked on this project.

References

[1] S. Skúli, “How to generate music using a lstm neural network in keras.” Towards Data Science, Dec 2017. [Online]. Available: https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5
[2] Zachary, “How i built a lo-fi hip-hop music generator,” Nov 2020. [Online]. Available: https://ai.plainenglish.io/building-a-lo-fi-hip-hop-generator-e24a005d0144
[3] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, “Midinet: A convolutional generative adversarial network for symbolic-domain music generation,” Jul 2017. [Online]. Available: https://arxiv.org/abs/1703.10847
[4] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” Nov 2017. [Online]. Available: https://arxiv.org/abs/1709.06298
[5] I. Tham and M. Kim, “Generating music using deep learning,” Aug 2021. [Online]. Available: https://towardsdatascience.com/generating-music-using-deep-learning-cb5843a9d55e
[6] [Online]. Available: https://salu133445.github.io/lakh-pianoroll-dataset/
[7] “Welcome!” [Online]. Available: http://millionsongdataset.com/
[8] “Midi music samples.” [Online]. Available: https://drive.google.com/drive/folders/158ZmhUffYX0FIQ0tKpqmT2DQFilD7B4s?usp=sharing