Aroona Atmaram

MSDS 458: Music Generation Using Deep Neural Network

June 01, 2022

Abstract

The use of deep learning to solve problems in literary arts has been a recent trend that has gained a lot of attention and automated generation of music has been an active area. With enough data and the correct algorithm, deep learning is able to create music that sounds human generated. This project outlines different experiments to music composition through Neural Network model using the ABC form of music notation.

Introduction

Deep Learning has improved many aspects of our lives, in ways both obvious and subtle. Though there is ongoing discussion around deep learning as a black box and the difficulty of training, there is huge potential for it in a wide variety of fields including medicine, virtual assistants, and eCommerce.

One fascinating area in which deep learning can play a role is at the intersection of art and technology. To explore this idea further via deep learning I want to propose building generative RNN models that create ABC notation sheet music with well-formed structure and stylistic conventions without predefining music composition rules for the models. Inspired by the research paper - Music Composition using Recurrent Neural Networks. Deep learning Methods help even novice music learners generate good quality music. I plan to use limit the experiments to single instrument to start with and later extend it to multiple instruments depending on the success of one instrument music generation. It is simpler and computational easier to use single instrument music.

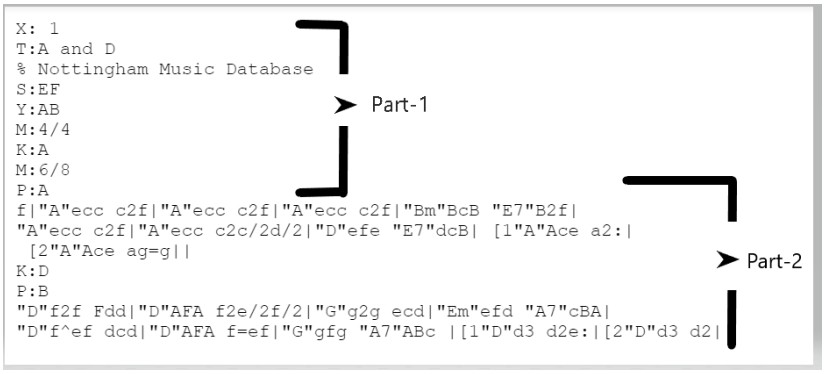
Literature Review

Generating rhyming and rhythmic patterns has also been a major focus. Recently, a”deep-rhyming” language model was created for rap lyric generation (Jones, M. 2017) and another paper focused on generating Chinese quatrains which also have strict rhyme and rhythmic schemes Wang, Z. (2017) Previous work on generation of Music comes from Karpathy in 2015 . In the article, he discusses the effectiveness of character RNNs in generating poems and dialogues.

Data

For our research the dataset is the ABC “Jigs tune” of the Nothingam Music Database. There are 340 tunes in the dataset with each with a metadata and ABC tune format stored in an `input.txt` file. The `figure-1` below shows a representation of a single tune from the `input.txt` file. The `Part-1` of the tune is its metadata and `Part-2` is the ABC-format of the tune itself.

`Figure- 1 – Data Preview`



The first step of preprocessing is to assign a numeric code to each musical characters and there are a total of 86 characters in the musical vocabulary. The `figure-2` below

shows the numeric representation of each character in the vocabulary that are stored in the `char_to_idx.json`.

Figure- 2 – Character encoding of the vocabulary


```
{"\n": 0, " ": 1, "!": 2, "\"": 3, "#": 4, "%": 5, "&": 6, "'": 7, "(": 8, ")": 9, "+": 10, ",": 11, "-": 12, ".":
13, "/": 14, "0": 15, "1": 16, "2": 17, "3": 18, "4": 19, "5": 20, "6": 21, "7": 22, "8": 23, "9": 24, ":": 25,
"=": 26, "?": 27, "A": 28, "B": 29, "C": 30, "D": 31, "E": 32, "F": 33, "G": 34, "H": 35, "I": 36, "J": 37, "K":
38, "L": 39, "M": 40, "N": 41, "O": 42, "P": 43, "Q": 44, "R": 45, "S": 46, "T": 47, "U": 48, "V": 49, "W": 50,
"X": 51, "Y": 52, "[": 53, "\\": 54, "]": 55, "^": 56, "_": 57, "a": 58, "b": 59, "c": 60, "d": 61, "e": 62, "f":
63, "g": 64, "h": 65, "i": 66, "j": 67, "k": 68, "l": 69, "m": 70, "n": 71, "o": 72, "p": 73, "q": 74, "r": 75,
"s": 76, "t": 77, "u": 78, "v": 79, "w": 80, "x": 81, "y": 82, "z": 83, "|": 84, "~": 85}
```


Model

Recurrent Neural Networks (RNNs), different from the feedforward neural networks, allow for incorporating long term dependency in the model. Theoretically, it can remember infinitely long sequences, although in practice it is limited by the vanishing gradient problem (Hochreiter et al., 2001). During the training of back-propagation through time, the gradient is extremely diminished by multiplications of sigmoid operations.

LSTM (Long Short-Term Memory) units solved this vanishing gradient problem. LSTM allows the gradient to be flowed by a separate path with not multiplication but addition operations. It can remember the context information for a relatively long time, adapted to data in time sequences. Given a list of data in time sequences X = {Xt |t=1 T}, in every time step, there is an input vector Xt ∈ X. We suppose that these t time steps are logical in some extents, like characters in language models, and need to learn their potential logical relation, the relationship of the context. Since every node in LSTM has a memory cell, for LSTM, the output vector yt at the t time step depends not only the input xi , but the input of previous t-1 time steps, $x1$, …, $xi-1$, whose effects on yt are decided by the implicit state $ht-1$. Through the mechanism of

information flowing controlled by input gates and forget gate, the memory cells in LSTM units need to remember or forget some information.

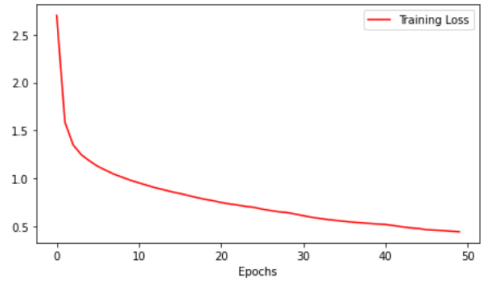
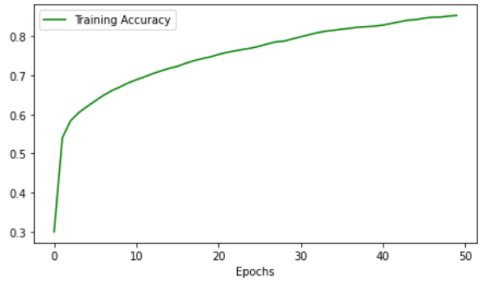
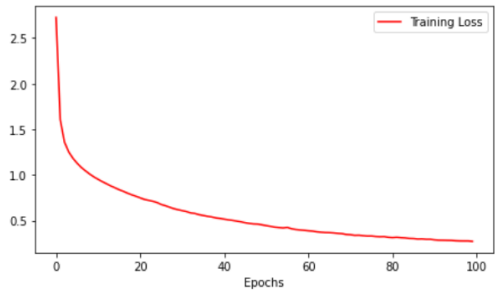
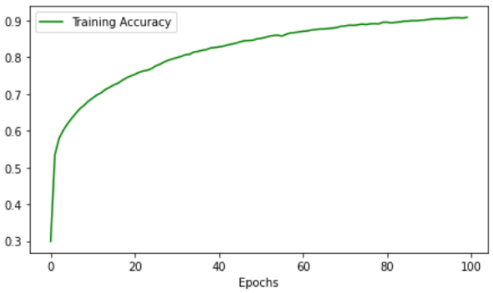
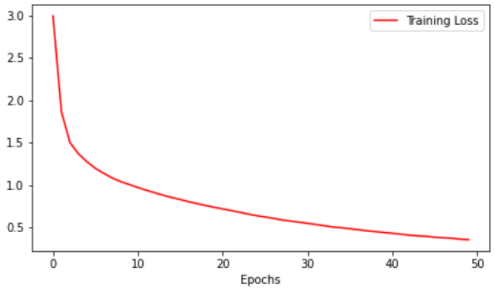
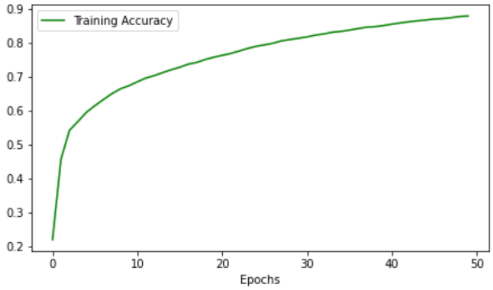
Experiments of different Model for Music Generation

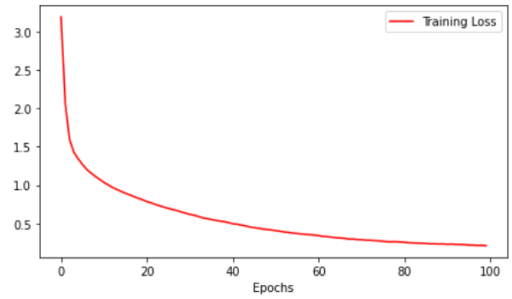
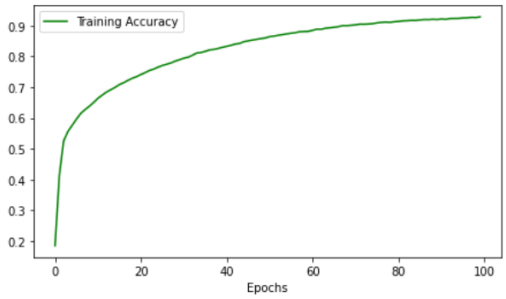
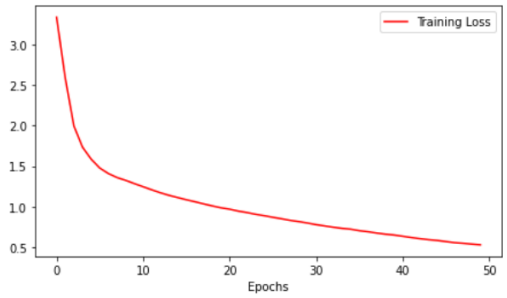
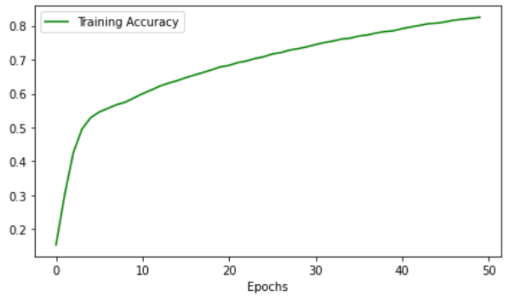
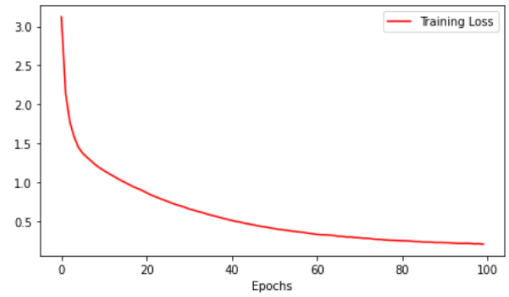
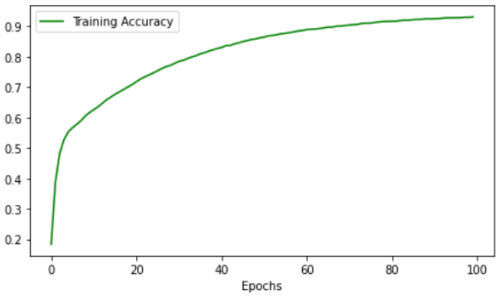
The model is a sequential model with embedding layer as the first layer which takes the input of the vocabulary of size 86 and returns an output of dense vector that is of size 512. The embedding layers are extremely useful when we have text data with batch operating especially in NLP tasks. Next we experiment with 1, 2 and 3 LSTM layers and for each LSTM layer there is a drop layer added. The dataset we are working with a vocabulary size of 86 and total length if a few hundred thousand only. The dropout is added primarily to avoid the situation of over fitting the model. After the LSTM layer a Time distributed layer with a 'softmax' activation is added. The output is vector of size 86 that is one hot encoded. The output is analogous to idea of having 86 classes with only one value set to 1 and the rest set to 0. We use Stochastic Gradient Descent with a Batch size of 64 for training the models.

The model takes an input model and generated 86 probabilities corresponding to the 86 characters we have of the occurrence of the next model. The model takes an input of the number of epoch, seed character and the length of the sequence we intent to generate.

Model	# of LSTM Layers	# of Epochs	Batch Size	Regularization	# of trainable Parameters
1	1	50	64	Dropout	825906
2	1	100	64	Dropout	825906
3	2	50	64	Dropout	1351218
4	2	100	64	Dropout	1351218
5	3	50	64	Dropout	1876530
6	3	100	64	Dropout	1876530

Results

Model	Loss	Accuracy	Time in sec	Generated Music
1	0.57	0.84	100.116	Model1 50 Epochs
 				
2	0.27	0.90	195.34	
 				Model1 100 Epochs
3	0.35	0.87	111.01	
 				Model2 50 Epochs
4	0.21	0.92	216.09	

 				Model2 100 Epochs
5	0.53	0.82	133.72	
 				Model3 50 Epochs
6	0.20	0.93	265.91	
 				Model3 100 Epochs

Analysis of results

All models have above 80% accuracy with the best training accuracy of 93% and loss of .20 of Model 6. Rating the music generated qualitatively, I would pick Model 6 as the best model and model 4 as the second best. The music generated by model 6 had better finishing notes. When considering training time, Model 4 provided good quality music with a minute lesser training time compared to Model 6.

Conclusion

The patterns will be different for every generation but similar to the training data. These melodies can be used for a wide variety of purposes- to enhance artists' creativity through inspiration, as a productivity tool to develop new ideas, as additional tunes to artists' compositions, and draw inspiration to complete an unfinished piece of work.

However, this model can still be improved. Our training data consisted of a single instrument, the piano. We could enhance our training data by adding music from multiple instruments. Another would be to increase the genres of music, their rhythms, and their timing signatures.

References

Swarbrickjones, Swarbrickjones, Eido on November 7, and Justine V on June 28. "DeepRhyme (D-Prime) – Generating Dope Rhymes with Deep Learning." Mike Swarbrick Jones' Blog, November 24, 2016. https://swarbrickjones.wordpress.com/2016/11/07/deeprhyme-d-prime-generating-dope-rhymes-with-deep-learning/

"ArXiv:1610.09889v2 [Cs.CL] 7 Dec 2016." Accessed May 31, 2022. https://arxiv.org/pdf/1610.09889.pdf

"The Story of Making a Music Generation Neural Network." linuxtut.com, June 30, 2017. https://linuxtut.com/en/6a433068534d57fb3b52/

Bodenhofer, Verena. Institute of Bioinformatics. Accessed May 31, 2022. http://bioinf.jku.at/publications/