Music generation by Deep Learning

Kushagra Gupta
Computer Science and Engineering
Medi-Caps University, Indore, India
kushagragupta0612@gmail.com

Milan Gupta
Computer Science and Engineering
Medi-Caps University, Indore, India
milangupta1998@gmail.com

Mohit Kala
Computer Science and Engineering
Medi-Caps University, Indore, India
kalamohit98@gmail.com

Binod Kumar Mishra
Assistant Professor, Dept. of Computer Science and Engineering
Medi-Caps University, Indore, India
bkmishra21@gmail.com

Abstract- Music is known to be a strong subject of research and entertainment and it has come a long way since the ancient times. It is evident that technology can automatically generate a melody or a suite with minimal human intervention. Artificial or Fabricated Music can easily be forged thanks to advanced research and hardware/software(s) which can be put into practice to alter and/or compose music for tranquil pleasure or motivation. This paper zooms in on Music, its elements, and the work done prior to this paper in the same domain.

The paper summarizes several feasible techniques for music generation; and in deep, different Deep Learning approaches namely Recurrent Neural Network (RNN)[1] and Long Short Term Memory (LSTM)[2], either of which can be used to train the model on Music files present in ".txt" format and in "abc"[3] notation. The "abc" format is a machine comprehensible representation of musical notes. The paper will mainly focus on LSTM to generate musical sequences.

After pre-processing the training data, the multi-layered neural network[4] will intake input in ".txt" format and will generate a unique musical sequence based on user input similar to the genre of the training dataset. To save time we can load weights from previous model training to get a desired output.

This sequence is then converted to a compressed file format for audio which is called a MIDI[5] file and can be directly played on any audio player.

The paper also briefs about the performance of LSTM model and a thorough analysis will be conducted on the model of training and its comparison to other model available for use.

Important Terms- **Music, Neural Networks, RNN, LSTM, "abc" notation, MIDI**

I- INTRODUCTION

Music is an organized and intentional arrangement of sound. It can be modified to be pleasant, coarse or dramatic. It is an art which requires precision to utmost extent. The notes are generated by composition and unity of the different pitch of musical notes. The single unit of the music is a musical note. It can have a single line of melody or may consist of multiple ones. Music consists of data that may resemble a repetitive pattern and this pattern decides the genre and suitable tempo of a particular suite to play on is trained with a sound of different frequencies and weights.

The motivation of building an AI[6] model to generate music is taken from art performers in history like Bach, Mozart and Beethoven. These performers fined tuned the art of playing piano which involves high precision and a bit of pattern understanding too. A good piece can be composed by working with Fibonacci numbers and creating good chords to match the tune.
Music can be generated by utilizing existing algorithms via the genetic approach. For the project stated, a Genetic Algorithm(GA)[7] can be used which will sacrifice the training time as it is an algorithm is an advanced searching technique used in computation industry to get exact or near approximate number of solutions for optimization or search problems. On the other hand, a neural network is a very complex non-linear statistical data modeling tool that can be used to model the complex relationships between single or multiple inputs and outputs and/or find various patterns in data in the meantime.

Being popular and easy to use, the key aspect of the project is to use LSTM network for easily training a model on a particular data-set and can make some sense out of it by understanding how the data is related to each other and how the prediction will work. We just have to map the inputs to the forthcoming layers and them, to the final output layer.

The organization of the paper is as follows: Section II provides a description of already attempted work in the music sector. Section III involves the theory part involved along and addresses the challenges too. The working and implementation of the deep neural network and soft-wares are mentioned in Section IV. Section V briefs in the conclusive part of the model for music generation and narrows in the comparison with earlier work in the field. Section VI contains the Conclusions, Acknowledgement and References.

II - RELATED WORK

The previous automatic music generation systems have been quite popular because of the hype and efficiency that they lived up to, and they vary from dedicated research resulting in open source soft-wares to enterprise applications.

Latter work related to the music modeling, polyphonic music and notable round time series probability density estimation(centered), has attained remarkable feat. Also to mention, the RNN when combined with Restricted Boltzmann Machines (RNNRBM)[8] and other recurrent energy based models has gained some attention. "Deep Learning for Music"[9] is a paper written by Huang and Raymond who take note of the above stated information and wrote a paper on a generative model to perform end-to-end learning and generation of multiple melodies using only deep neural nets.

Another motivation for the paper is "Generating music using LSTM network"[10] by Nikhil Kotecha which addresses the use of Bi-Axial LSTM[11] network trained with kernel reminiscent with convolutional kernel.

After various researches conducted in the field different organizations and individuals have come up with different ideas and network architectures. Some related work and honorable mentions are DeepBach[12], GANsynth[13] and Coconet[14]. These ML models use original data to work on and generate notes in chronological order from beginning to the end. Be it Bach or any other composer, these networks are flexible to support the compositional process.

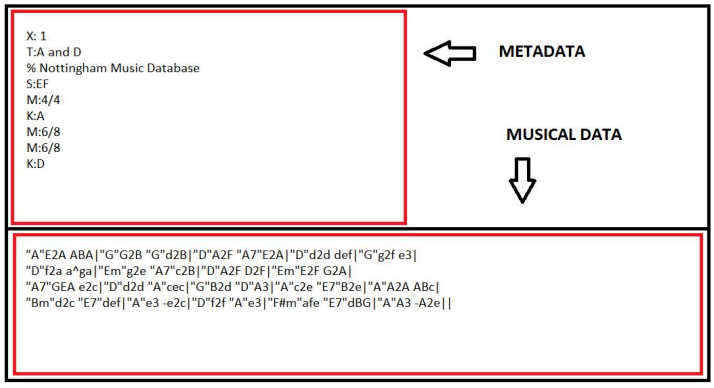
Most of the networks use classic and jazz music as they are present in free-form for training. For this format of music working in time and spatially structured data CNN[15] and LSTMs are a good when the scope of prediction is low and the network does not need the context of the entire song at the same time. For much complex situations one can always use multiple auto-encoders and decoders along with Principal Component Analysis.
There are many approaches to train a model when it comes to complex or jumbled data. The model can be tuned or improved based on the output requirements. As mentioned earlier, we can experiment with different network units to check the output of the various networks and compare the efficiency later on. At least for now we will be using charRNN technique in the project that has been undertaken.

III- METHOLODOGY

Our model takes general inspiration from both the described papers but our music training data not being that massive, follows generic LSTM approach with 3 different layers and around 90 epochs for appropriate prediction efficiency. The methodology described in the papers we inferred inspiration from , either use a Bi-axial LSTM model or a combination of many LSTMs and RNN layers to achieve the desired output. The papers used bi-axial LSTMs or combination to learn and predict the output at the same time.

A major obstacle standing in the way is resolved by data pre-processing by converting the data into tokens to reduce the training time and increase the efficiency of the throughput. Here, after training when the prediction is done on the selective user outputs, the output is obtained in the "abc" notation and to interpret it into a user-comprehensible format, the file needs to be in a specific format which is MIDI. The general difference between MIDI and "abc" format for selection is the format of representation of the data, meta-data and usability. We have used Colin Hume's "ABC TO MIDI"[16] for the conversion purpose.

The data that we used was from Nottingham Music Database[17] which used an "abc" shorthand form of musical notation present in the ".txt" file on which our multi-layered LSTM model was trained. The data-set contains the music data of multiple genres and artists.



ABC Notation of music

Fig 1: ABC representation of musical notes

The "abc" format contains of 2 major sections. The metadata that represents the information about music title, author, tune number, rhythm, key, note length etc. and the next section where information is represented in the notes which are in turn represented in the alphanumeric form.
Another unanswered question is of the selection of LSTMs over RNNs in the training phase. The RNNs suffer from a significant problem known as the vanishing gradient problem. It is comparatively uneasy for a traditional RNN network to maintain long term learning temporal dependency. Even considering this into account we had an option to use GRUs as well but again, taking into account that multi-layered LSTM was feasible considering the data size and prediction scope, there was no need to complexify the structure we were using already.

IV- IMPLEMENTATION

This section explains the whereabouts of the model used to generate a random suite based on the previous samples it saw during training. The training and data was remotely handled on the local machine and the same can be replicated on platforms like Kaggle or Google Colaboratory, with essential machine learning and deep learning libraries installed as per requirements. The basic deep learning library used was Keras which uses Tensorflow in the back-end.

A. The Neural Network Architecture-

Out data-set mostly consisted of the homophonic data(similar or predominating musical texture) and the LSTM layers are trained to understand and improvise on the knowledge base of relation and occurrence of a particular note. The network tries to figure out the probability of occurrence of every different note after a particular note and when it does, the network updates the knowledge and does the same for all the data it has got for training.

To attain the goal we use a variation of RNN also known as char-RNN[18]. The idea is to feed the network one character at a time and out data-set has 87 such unique character which can be sent sequentially for the network to understand the appearance pattern of various characters after one another. Our model uses Many-to-Many RNN(Fig 4) in which the number of inputs and outputs are the same.

In basic terms, we need to predict multiple characters and thus this problem can be classified into multi-class classification problem. For this problem we also need to choose some specific optimization functions and loss functions. One loss function that we have used is a "Categorical Cross-Entropy"[19] which is also known as the "Multi-Class Log Loss". The last layer comprises of Soft-max activation units and the number will be the same as the number of unique characters in the data that we are training the model on. Soft-max function solves the problem of assigning an instance to respective class when we have more than one class in the data-set. Our RNN network can be a LSTM as well and the training can be done by back-propogation while iterating the process and using "Adam" optimizer.

B. RNN and data training-

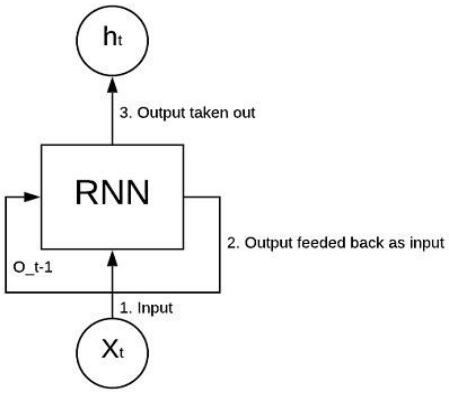


Fig 2: RNN

The image above describes a RNN unit and how the characters fed one character at a time. Xt is a single character which is thrown in the network at time interval 't' that is given as an input to a RNN unit. And as of O_t-1, it is resultant of the 't-1' character which acts as an input to the RNN unit at time some 't'. Thus output 'ht' is generated. Again, 'ht' will act as an input to RNN as 'O_t-1' in the upcoming time step.

'ht' will act as a next character in the sequence. If the music data is represented in a sequential format [a, b, c, a, d, f, e,…] then , the first character is sent into the network and b is generated as the resultant output. The sequence will go on till the last character which when fed into the network, will generate the first-most character ex: "e-> a". The RNN should output the upcoming character in sequential manner. In this process the network learns the sequential associativity between characters and learns to generate the output.

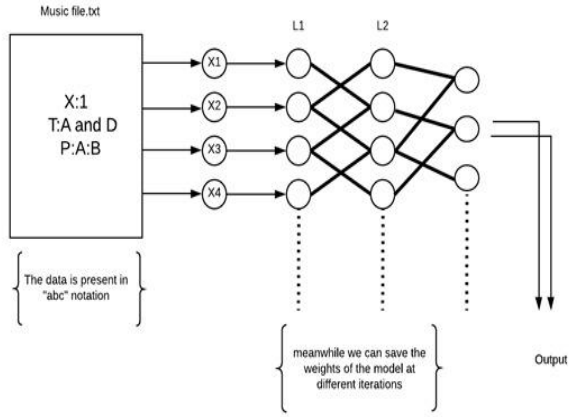


Fig 3: Working and parsing of the data in the network

Fig 3 describes the data input and output in network and Fig 4 describes the Layer-wise network distribution with the underlying softmax units.

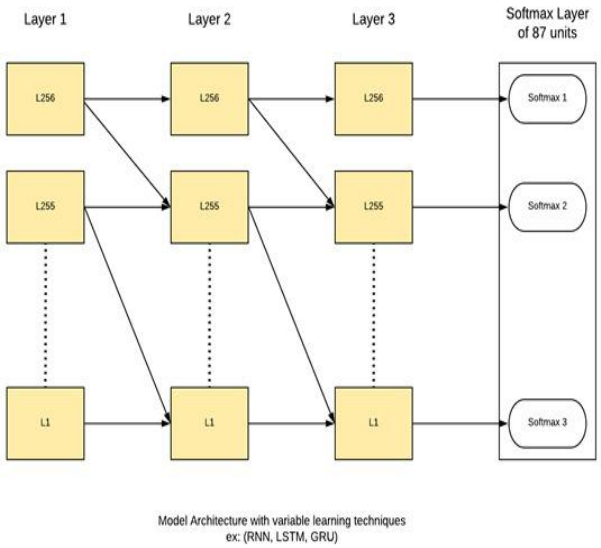


Fig 4: Layer-Wise architecture of the network with activation function

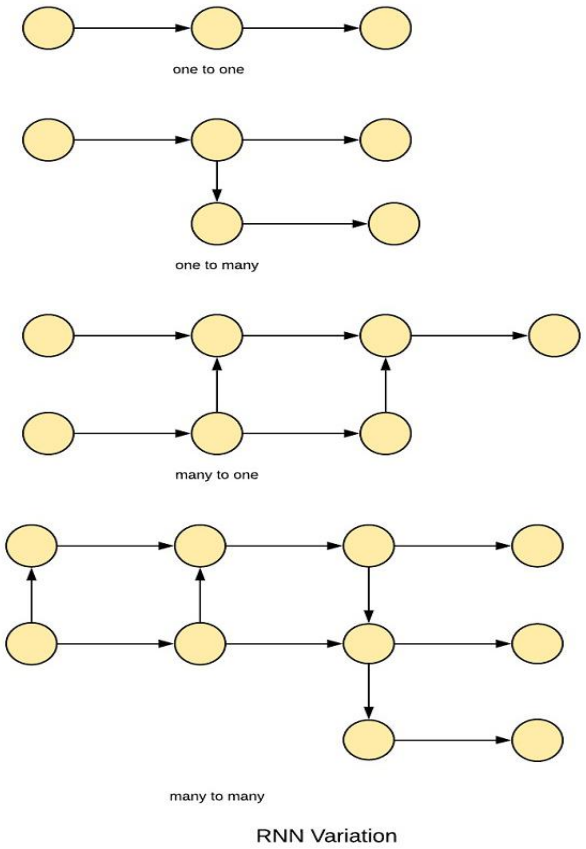


Fig 5: It shows variation of cells within the RNN layer. The layers can be modified with altering the basic structure of network by either increasing the number of neurons or decreasing the neurons and the optimizers can also be altered for understanding the variation in the network and the output generated later on.

C. Design Concept and structure of the system-

The first phase is the conversion of "abc" text format to an encoded format so that it can be passed in the network itself. We have to preprocess data carefully and clean it thoroughly. For training, we will have to transform the data into batches, and we have selected batch size till 16 and the sequence length till 64. The general idea is to send 16 batches of 64 sequences of data for training. After getting all unique characters from the data we assign a token to every unique character. Then we store them in token value pair in a dictionary.

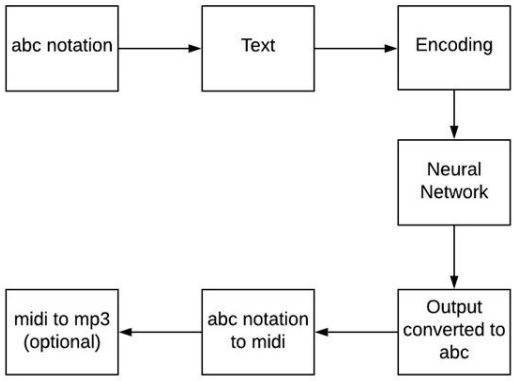


Fig 6: Conceptual architecture of the model

The next phase of network architecture is the training part and the general sequence of model is described in Fig7 shown below.

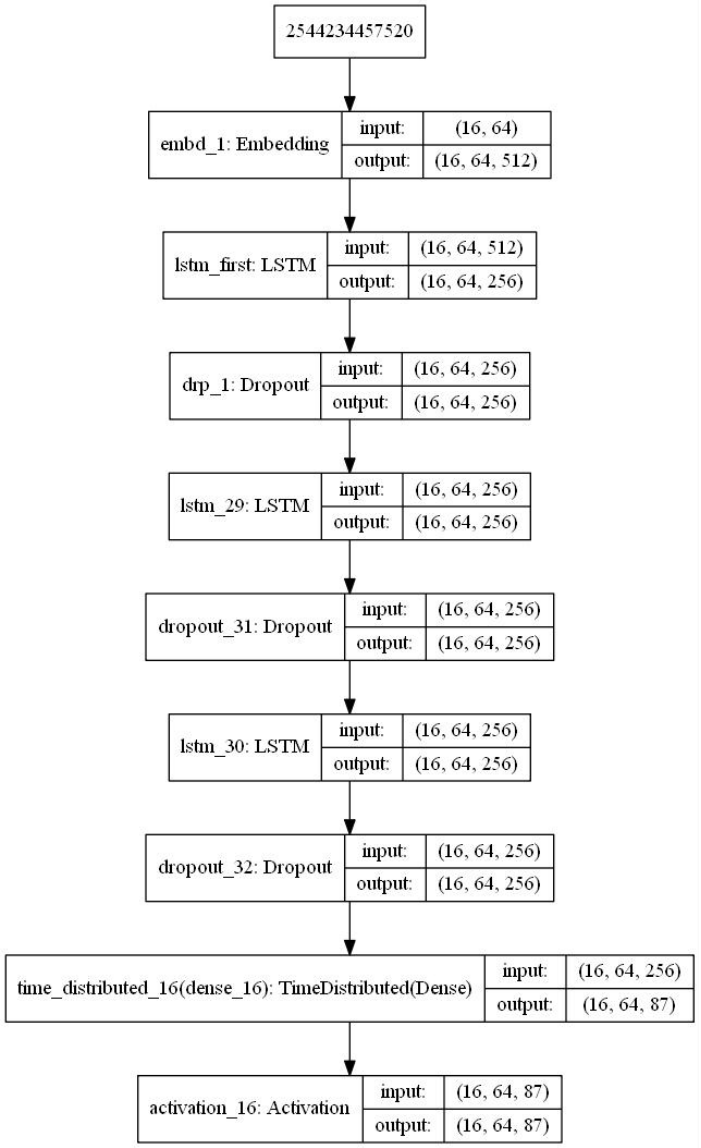


Fig 7: The sequential Model for training

The sequential model that we have used here contains of an Embedding layer at the start to integer encode the data. The next six layers consist of three simultaneous 256 units of LSTM units in a layer and Dropout layers ready for intake of data in the batches, and sequences defined already. The next layer is a Time Distributed Dense layer which is responsible for making the dense nodes of the layers identical so that the weights and the bias of all layers remain the same for the entire time of training.

And finally we have a softmax activation function associated at the end which is responsible for multi-class data instance assignment as described in Part A.

We have to make sure that the "return_sequences" parameter is set as TRUE as we have to generate output for every character at each time step. We require the network to create an output for a particular character when as an input for the next character and hence learn in the process. After the training, the data that, the output we get is still present in the "abc" format, so we have to

convert it to a computer comprehensible midi file format to understand which is small and has near to none data loss.

V- RESULTS

The training time required to train the model at 90 epochs is around one and a half hour on local machine. To capture most efficient training time we can increase the learning rate, normalize the data in an efficient manner, lessen the model complexity or use a powerful GPU or increase the batch sizes. The resultant loss that we obtained while training the model is reduced to 0.18 which was around 1.33 since the start of training.

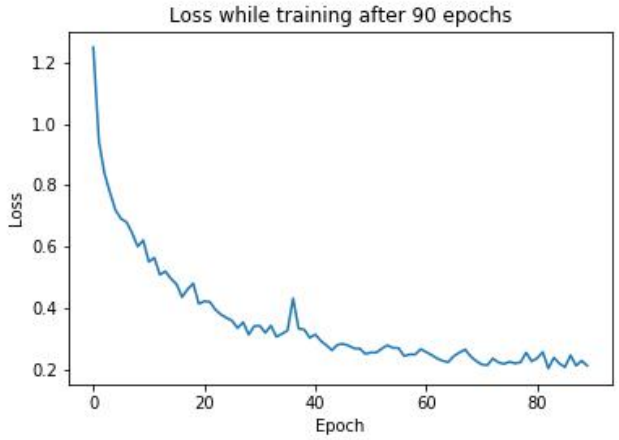


Fig 8: The loss while training the model

The accuracy of the model improved with the increase of epochs. Sometime in between we encountered the stagnant progress, and then we had to play with the learning rate and tune some other parameters as well. Starting from 58 percent training accuracy at the first iteration, we find the model perform uniformly near around 70 epochs and thus the training time can also be shortened after altering the number of epochs to certain limit depending on the accuracy. Final training accuracy obtained after 90 iterations was near around 93.22 and can be improved further after increasing the epochs, model complexity or training time.

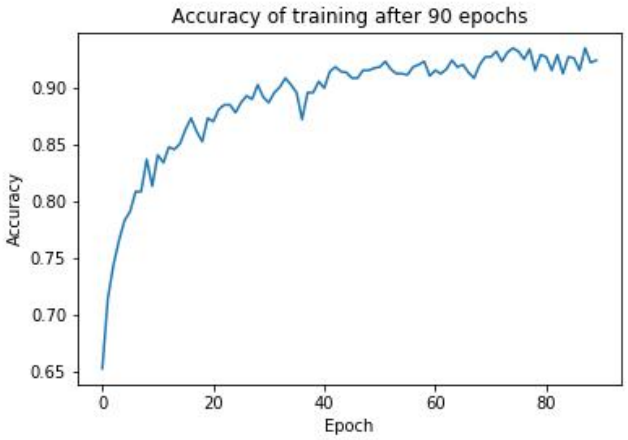


Fig 9: The accuracy while training the model

We can observe a certain inverse relation of training accuracy and loss in the 2 graphs depicted in Fig 8 and Fig 9.

```
1. Which epoch number weight you want to load into the model(10, 20, 30, .
sic: 90

2. Enter any number between 0 to 86 which will be given as initial charcte

3. Enter the length of music sequence you want to generate. Typical numbe
ardly generate any sequence: 500

MUSIC SEQUENCE GENERATED:

K:G
"G"G3 GAB|"C"c3 cde|G2E A2E|"C"G^FG A2G|[1"G7"GAG FGA|
"G"BdG BAG|"D"FED D3|"G"GAB DGB|"C"AcB "Am"AGE|
"G"GBd gbg|"C"edc "G7"dBG|"D7"FGA DEF|"G"G3 G2||
P:B
V:1
|:d|"Gm"g2G GFG|"G7"GAB e2G|"C"eGg e2c|"Dm"def "G7"g2a|
"C"gec "E7"de^g|"Am"a2A "D7"A2G|"G"GAG "C"g2e|"G"dBG "D"A2B|
"G"G3 "C"gfe|"G"dcB "Em"deg|"Em"fed "D7"e2d|
"G"gba "C"gfe|"G"d3 "D7"d3|"G"G3 "C"g2e|"G"dBG "D7"A2F|"G"G3 -G2:|
```

Fig 10: Output snapshot of a user query

V- CONCLUSION

In conclusion, the algorithm that we tested and implemented, the LSTM RNN model performs with a good 93 percent accuracy. Largely the good accuracy of the output was due to the encoding process and for the encoder layer and due to the tokenization of the data, the character sequence prediction became quite easy. The model can simultaneously be used with other alternatives to LSTMs and the prediction of the homophonic data can be increased by taking note of the pre-processing techniques and iterations. One has to make sure of the avoidance of over-fitting of data which may reduce the randomness of prediction by great extent. Taken into account the musical notes we borrowed from the Nottingham music data-set for training the model , the data can be increased for improving the training accuracy and quality output as well.

V- ACKNOWLEDGEMENT

This research was backed by Medi-Caps University. We thank the Assistant Professor Mr. Binod Kumar Mishra, Medicaps University for their expertise and assistance in all aspects of our study and for their assistance in writing the paper. Although any error that may have crept in is the doing of our own and shall not be and blamed upon the references or people externally included in the project.

VI- REFERENCES

[1] Sherstinsky, A., 2020. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, p.132306.
[2] Staudemeyer, Ralf & Morris, Eric. (2019). Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks.
[3] Mansfield, S. (2020). How to interpret abc music notation.
[4] Svozil, Daniel & Kvasnicka, Vladimir & Pospíchal, Jiří. (1997). Introduction to multi-layer feed-forward neural networks. Chemometrics and Intelligent Laboratory Systems. 39. 43-62. 10.1016/S0169-7439(97)00061-0.
[5] de Oliveira, Hélio & Oliveira, Raimundo. (2017). Understanding MIDI: A Painless Tutorial on Midi Format.

[6] Miller, Tim. (2017). Explanation in Artificial Intelligence: Insights from the Social Sciences. Artificial Intelligence. 267. 10.1016/j.artint.2018.07.007.
[7] Man, K.F. & Tang, Wallace K.s & Kwong, Sam. (1996). Genetic algorithms: Concepts and applications. Industrial Electronics, IEEE Transactions on. 43. 519 - 534. 10.1109/41.538609.
[8] Montufar, Guido. (2018). Restricted Boltzmann Machines: Introduction and Review.
[9] Huang, Allen & Wu, Raymond. (2016). Deep Learning for Music.
[10] Kotecha, Nikhil & Young, Paul. (2018). Generating Music using an LSTM Network.
[11] Mao, Huanru & Shin, Taylor & Cottrell, Garrison. (2018). DeepJ: Style-Specific Music Generation.
[12] Hadjeres, Gaëtan & Pachet, Francois & Nielsen, Frank. (2017). DEEPBACH: A STEERABLE MODEL FOR BACH CHORALES GENERATION.
[13] Engel, Jesse & Agrawal, Kumar & Chen, Shuo & Gulrajani, Ishaan & Donahue, Chris & Roberts, Adam. (2019). GANSynth: Adversarial Neural Audio Synthesis.
[14] Huang, Cheng-Zhi & Hawthorne, Curtis & Roberts, Adam & Dinculescu, Monica & Wexler, James & Hong, Leon & Howcroft, Jacob. (2019). The Bach Doodle: Approachable music composition with machine learning at scale.
[15] Jianxin Wu (2020). "Convolutional neural networks".
[16] (2020). Retrieved 10 April 2020, from https://colinhume.com/music.aspx)
[17] (2020). Retrieved 10 April 2020, from https://colinhume.com/music.aspx)
[18] Kim, Yoon & Jernite, Yacine & Sontag, David & Rush, Alexander. (2015). Character-Aware Neural Language Models.
[19] Zhang, Zhilu & Sabuncu, Mert. (2018). Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels.