LYRICS TO MUSIC GENERATOR: STATISTICAL APPROACH

V.N Aditya Datta Chivukula[1], Abhiram Reddy Cholleti[2] and Rakesh Chandra Balabantaray[1]

[1]Department of Computer Science and Engineering
[2]Department of Electronics and Telecom Engineering
International Institute of Information Technology, Bhubaneswar, Odisha, India

ABSTRACT

Natural Language Processing is in growing demand with recent developments. This Generator model is one such example of a music generation system conditioned on lyrics. The model proposed has been tested on songs having lyrics written only in English, but the idea can be generalized to various languages. This paper's objective is to mainly explain how one can create a music generator using statistical machine learning methods. This paper also explains how effectively outputs can be formulated, which are the music signals as they are million sized over a short period frame. The parameters mentioned in the paper only serve an explanatory purpose. This paper discusses the effective statistical formulation of output thereby decreasing the vast amount of estimation of output parameters, and how to reconstruct the audio signals from predicted parameters by using 'phase-shift algorithm'.

KEYWORDS

Audio Signal Processing, Statistical Machine Learning.

1. INTRODUCTION

Natural language processing algorithms are very helpful in understanding human-computer interaction effectively. But it becomes necessary to try different approaches in managing the same problem as it enhances our understanding and may spark up many brilliant ideas in different fields. This makes overall research in the field effective. In recent times generator has become one of the most important fields of study because of many reasons including its capability in understanding human-like thinking in specific fields like music, literature, and so on. This also helps us in increasing the capability of capturing the complex features and reciprocating them till the output. Hence, any generator algorithm can become very important in understanding various applications and details on which the basic functionalities are dependent. Hence, it is definitely a growing field in terms of research and application of which we tried to propose a new approach towards understanding it.

2. Literature Review

Lyrics to music generators is a certain application that is in its nascent stage and yet has overwhelming demand in the research application field. Currently, the most optimum results are published by jukebox [1]which is a project by OpenAI. In this work they used deep learning techniques where they actually fed the raw data without using any extraction techniques to separate voice from the original song, instead they used raw audio to train the model which spits out raw audio in return, where they had to use symbolic music for better results. They have concentrated more on working with VQ-VAE [2] model to compress raw audio data and then used autoregressive transformers [3]. The model actually learn the compressed audio generation which is generated in the form of discrete codes and it makes the process computationally expensive in the data pre-processing stage itself and has a risk of overfitting. They have used the attention model [4] mechanism extensively to learn the interdependencies between the lyrics and capture the similarities which are very risky as the dataset at the current stage is very less. The dataset the main model is learning is also an output of autoregressive transformers which is actually decreasing the originality of the information itself which may result in content tampering as a result of overfitting or underfitting. We propose a completely different approach and well-established facts and theory which is completely based on statistics during pre-processing of data and preserving the original information intact. We also propose the decoding part in a much simpler sense using a simple fact of superposition which works almost in every case without actually using deep learning approaches. Our propose here is a purely statistical approach right from encoding, learning parameters for which we used a regression algorithm [5], and finally, for the decoding part we have come up with a new algorithm that effectively works in the majority of cases.

3. Proposed Method

The different components of the proposed method along with the results are mentioned below in various subsections.

3.1. Voice Separation

We know that music is recorded on both left and right channels, and the vocals which are present in the music are mixed with different instruments and it becomes hard to remove them. We used Fast Fourier Transformation [6] to make a comparison between the left and right channels with the vocal and non-vocal part of that song. On experimentation, we found that the amplitude corresponding to vocals are present when the magnitude of FFT of the left channel sample is greater than 100 and that of the right channel sample is less than 400, if the vocals are in a low pitch and have no chorus in the background. Hence, if the value lies in that range we assigned the FFT value to be 0 for that sample. After taking the inverse Fourier transform of both the channels, we found that the voice was decent when subtracting those channels from each other but the noise was developed in the signal. On experimenting with an instrumental track, we compared the frequency domain of the resultant signal and instrumental signal, and found that noise ranges from 900Hz to 14000Hz (approx.). To remove noise we used the band-stop filter. Figure 1, represents the extracted signal in the time domain while Figure 2, represents the extracted signal in the frequency domain, we can observe that signals contributing inthe frequency range 900Hz to 14 KHz were zero.

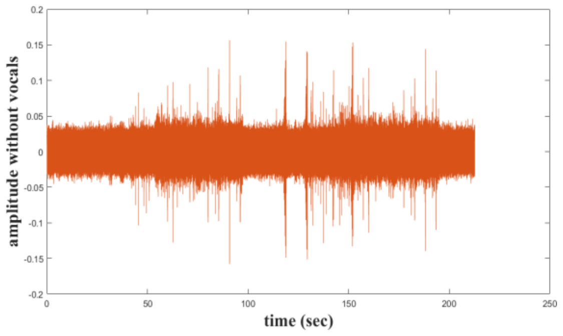


Figure 1. Amplitude Vs Time (Extracted Signal)

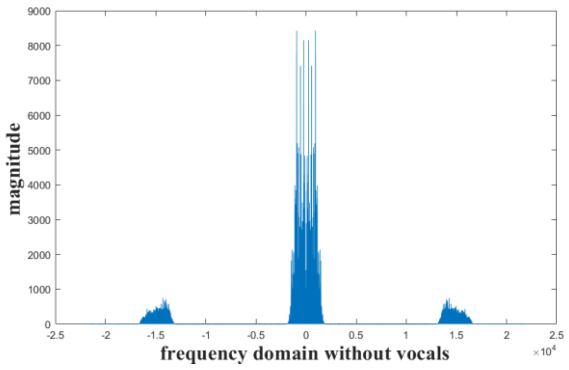


Figure 2. Extracted Signal in the frequency domain

On experimentation, we found that the angle between the music and its instrumental track is between 85-90 degrees. Figure 3, depicts the process of voice separation.

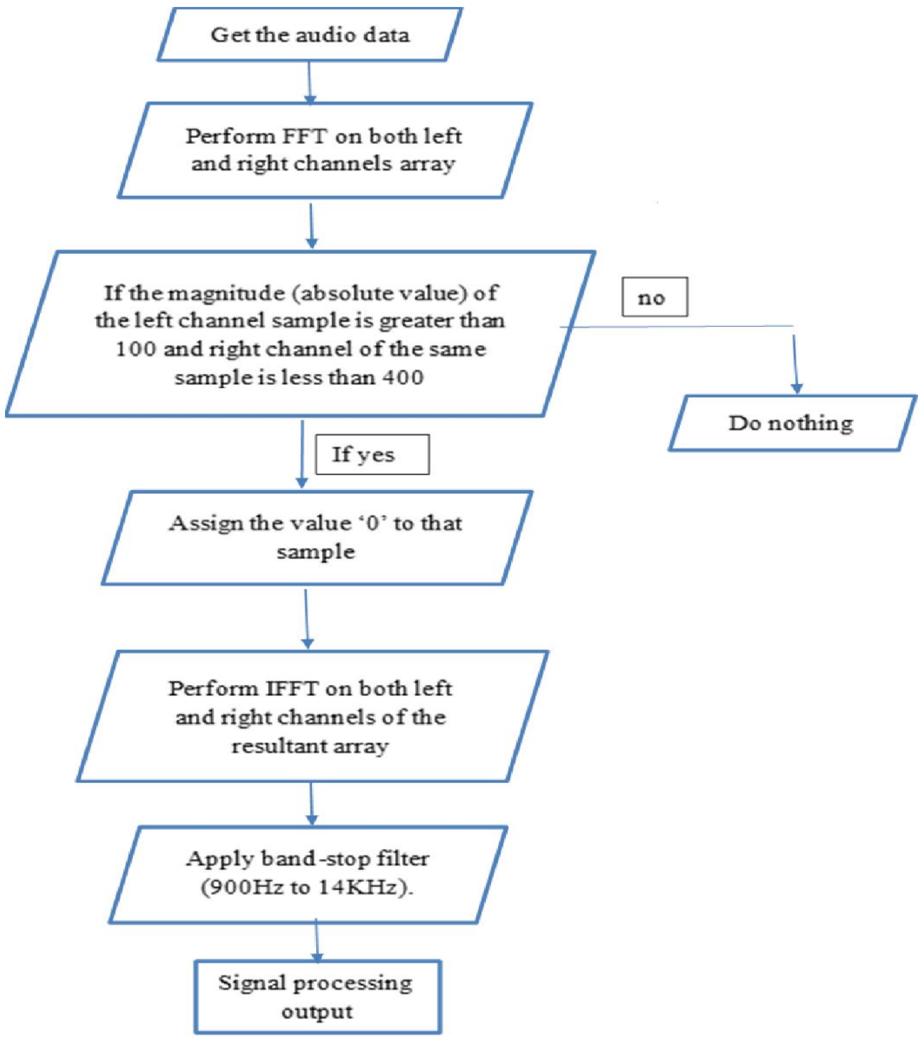

Get the audio data
Perform FFT on both left and right channels array
If the magnitude (absolute value) of the left channel sample is greater than 100 and right channel of the same sample is less than 400
no
Do nothing
If yes
Assign the value '0' to that sample
Perform IFFT on both left and right channels of the resultant array
Apply band-stop filter (900Hz to 14KHz).
Signal processing output

Figure 3. Flowchart of voice separation

3.2. Gaussian Representation

This subsection simply discusses the normalization of the audio array, One thing to be noted is that before normalization of the audio array, note down the mean and standard deviation of the array and label these parameters as O^1 and O^2 (O^1 and O^2 are two out of six output parameters we need to predict for test set). From the remaining four parameters, two parameters are the minimum and maximum value of the resultant array we get after performing normalization, which we are going to store for each sample of the audio dataset. Now, one has to just divide the absolute difference between the maximum and minimum value of the normalized array into 1000 intervals or more depending upon the range of minimum to a maximum value, while we were experimenting. We just selected 1000 intervals at random (as the number of intervals can be treated as a hyperparameter, one can always experiment with different values). Now, count how many values in the normalized array are falling under this specific interval and divide this count with the dimension or size of the normalized array. In this way, one can get the probabilities of each interval and if a graph is plotted taking the intervals on X-axis and corresponding probabilities on Y-axis, then, we get the approximate Gaussian curve as gaussian formula suggests one can get the Gaussian distribution [7] of n samples of data by converting the given data as follows

$$X_n^{|} = (X - \mu)/ (\sigma/ \sqrt{n}) \qquad (1)$$

where μ is the mean of the sample, σ is the standard deviation and 'n' is the sample size. For the implementation we have taken n=1 as there is no more than one song from the same distribution because one can observe each song's mean and standard deviation is different but, we will get an approximate gaussian as in Figure 4.

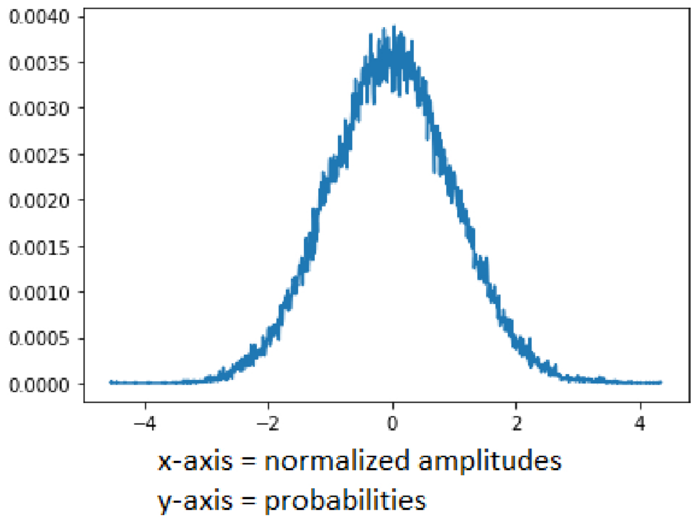


Figure 4. Depiction of approximate Gaussian

One, important feature or benefit one can get with this trick for music analysis is the enormous compression of data for the processing which decreases the data by (size of array-number of intervals) *100%. To understand this compression, if the size of the array is 10000000 and the number of intervals is 10000 then the compression, we have got is 1000 times less than the original which is, reducing the original to 0.1% of its size with minimal loss in information.

In Figure 5 one can see the values reconstructed by replacing every value of an interval with the mean of interval on the X-axis and original normalized array values on the Y-axis. The normalized array values and reconstructed array values are both in ascending order, so that, the graph represents only how close the reconstructed values are to the true normalized values without the arrangement of amplitudes which will be discussed later as to how to arrange these values in order close to the original normalized arrangement of amplitudes. Theentireprocess can be understood with the flowchart as shown in Figure 6.

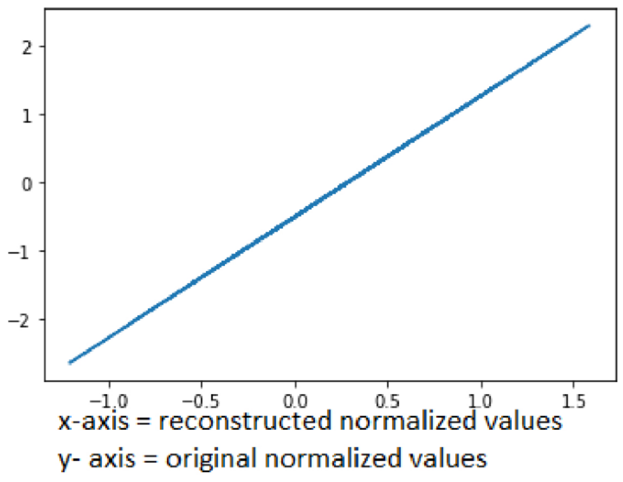


Figure 5. Depiction of similarity between Reconstructed: values and original normalized

3.3. Input Formulation

For designing the input, we have taken the lyrics and pre-processed it by performing sentence tokenization, word tokenization keeping in mind the chunking of characters which have expressional characters such as, ‘?’, ‘!’ etc, as they really influence the music. Because we assume that expressions are very much necessary in creating the music and one can never neglect the expression or feel or emotion in the lyrics. Then, we have taken the limit of 700 tokens for each song for applying the machine learning applications in the future. Every song is processed in the same way and made into a dataset of 100 songs each containing 700 tokens. Then, this dataset is passed on to the word embedding section where the word2vec library is used for converting the tokens into word vectors using a skip-gram model [8]. Then we get each token in a vector form, but we need a single value to represent the token but not a vector, and hence, one can just take the magnitude of the vector and embed the word token with its magnitude. Hence, the dataset can be reformulated with magnitudes of each token, if for a song the number of tokens is less than 700 then we can pad the remaining with zeros. Thus, the above input formulation can be completed. An important note is that the above-mentioned values are just for explanatory purpose and one can always try different deep learning and machine learning models to actually map the lyrics to the required output parameters. The value 700 is used here because there was no song that has more than 700 tokens, hence, this value acts as an upper bound for padding of sequences for input part training data. In Figure 7 there is a flowchart depicting the procedure followed for input formulation.

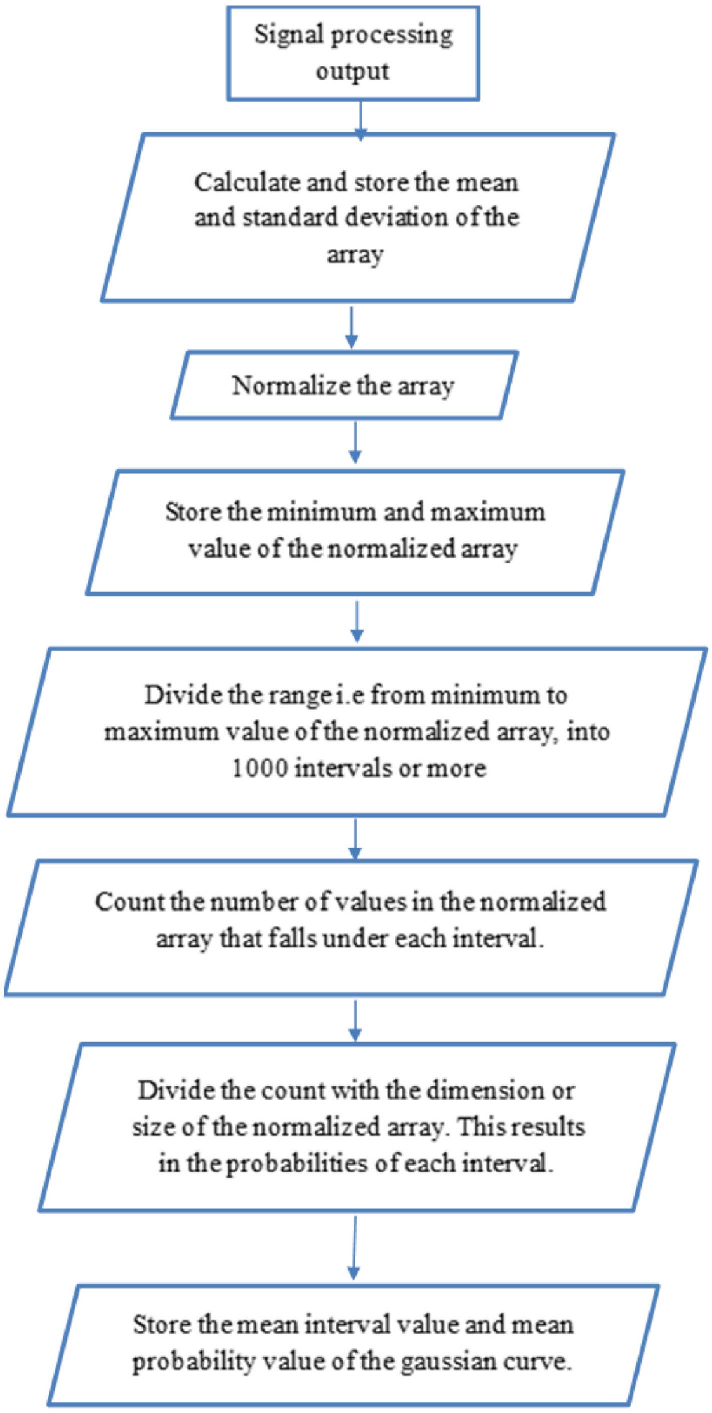

Signal processing output
Calculate and store the mean and standard deviation of the array
Normalize the array
Store the minimum and maximum value of the normalized array
Divide the range i.e from minimum to maximum value of the normalized array, into 1000 intervals or more
Count the number of values in the normalized array that falls under each interval.
Divide the count with the dimension or size of the normalized array. This results in the probabilities of each interval.
Store the mean interval value and mean probability value of the gaussian curve.

Figure 6. Flowchart of gaussian representation

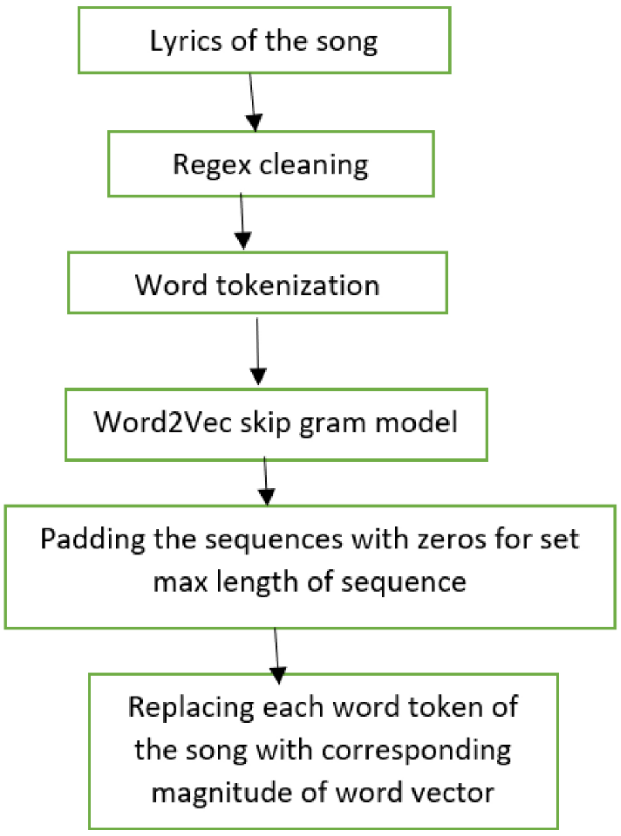

Lyrics of the song
Regex cleaning
Word tokenization
Word2Vec skip gram model
Padding the sequences with zeros for set max length of sequence
Replacing each word token of the song with corresponding magnitude of word vector

Figure 7. Flowchart depicting input formulation of the dataset

3.4. Output Formulation

As discussed in section 3.2, four out of six output parameters are mentioned and now the remaining two parameters are nothing but the mean frequency and mean probability, which implies, the total six parameters are:

Original array mean, original array standard deviation, normalized array minimum, normalized array maximum, gaussian mean interval, and gaussian mean probability. Each of the six output parameters is predicted by training them separately with six copies of linear regression models on the same input dataset as represented in Figure 8.

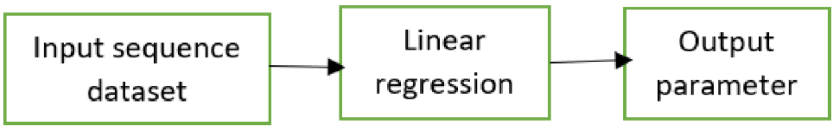


Figure 8. Block diagram depicting the training of machine learning algorithm

Once, one has obtained the parameters, first, they can take the mean interval and mean probability and calculate the standard deviation of predicted gaussian, then, take the mean and standard deviation of the predicted gaussian to actually construct the Gaussian curve and fix the X-axis range according to the minimum normalized array value and maximum value. Now, One can divide the range into a number of intervals they used while training for the experiment, we took 1000 intervals and calculated the average probability of each interval and store it in a 1000 sized array of probabilities. Finally, we just need to take an average interval value of 1000 intervals by which we get the average frequency array and repeat each value of frequency array into a new array of a size equivalent to normalized array size according to the corresponding probability of occurrence i.e if the first value in the probability array is 0.01 and average first interval value is -4.1 and also if the size of the normalized array used for training is 100000 array then the first 1000(0.01 * 100000) values of predicted normalized array should have -4.1 as its value and one has to repeat the same for remaining 999 intervals and corresponding probabilities. In this way, one can reconstruct the normalized predicted array, from this array taking the predictions of the original array to mean and original array standard deviation values we can reconstruct the original version of the array. But this original array will be in ascending order as we travelled the gaussian from left to right intervals. To tackle the original arrangement problem, we will discuss its solution in the arrangement section.

3.5. Predicting Outputs

Given input and output, one can test the results either by taking the deep learning approach or the statistical machine learning approach. When we tried to predict six output parameters each by using a simple neural network [9], CNN [10], RNN [11], and also LSTM [12], we did not get descent results of predictions and this may be because of the fact that dataset size is less. Therefore, we went for a statistical linear regression [13] technique by training each output with only one regression model for better predictions by which we used six copies of the linear regression model and six copies if same input data to train six output parameters individually and got decent values or predictions even on the low dataset. Hence, with this method we made the entire generator working completely on statistics. Just for example, In figure 10 we can see that some part of the graph is a straight-line relationship between the predicted amplitudes and original music amplitudes both arranged in ascending order, for a randomly selected test case. Hence, we can say that for a small dataset too we are able to capture the amplitudes reasonably well. We have taken 70 songs in training and 6 songs for testing and predicted the results, out of which is a predicted gaussian in Figure 9.

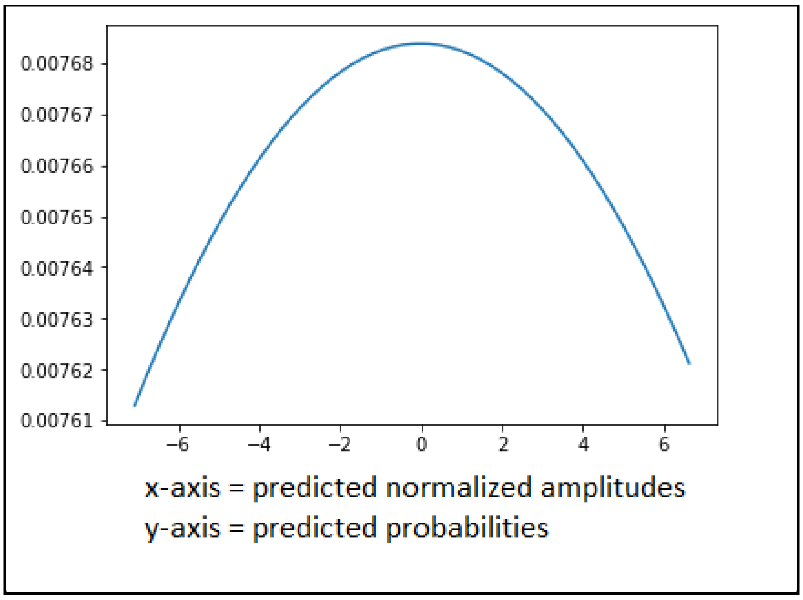


Figure 9. Plot depicting predicted normalized amplitudes (x-axis) and corresponding probabilities (y-axis)

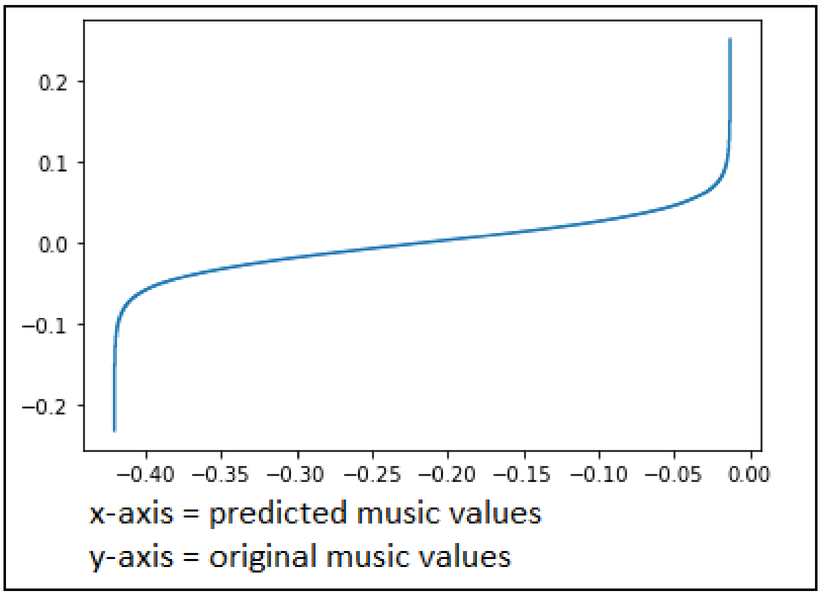


Figure 10. Depiction of similarity between original music values and predicted values both taken in ascending order to compare frequencies

3.6. Arrangements

The gaussian representation only helps us to predict the amplitudes accurately but misses in capturing the most crucial part of audio data which is the pattern of amplitudes or position at which a particular amplitude should occur in the output audio array. Because when the array is reconstructed from six output parameters, the array of amplitudes will be in ascending order but not in the desired order of amplitudes. Hence, we have arranged the amplitudes from ascending order into the desired order. On experimentation, we found that input normalized arrays are perpendicular to corresponding input normalized arrays in ascending order. So, the task finally is to rotate an n-dimensional array or vector by 90 degrees. To do this, one can follow the most obvious approach which is:

PERMUTATION METHOD: This method is very simple, just take all permutations of the array and check which permutation is perpendicular to the ascending order version of the array, but, the complexity of this approach is O (n!) which is very bad, especially dealing with music applications where array size is in millions.

3.7. Phase Shift Algorithm

This is a new algorithm to rotate a vector by some angle if that angle is possible with given values of the array. For our case, the angle is 90 degrees. This algorithm tries to find out the permutation of the vector which is 90 degrees to the ascending order array and if a 90-degree combination is not possible it gives the combination which is close to 90 degrees.

If predicted_array = [1, 2, 3] and we are required to find a near 90-degree combination. Hence, we follow the following steps:

Take int = [1] and fix its order of values.

Add next dimension value of predicted_array to init and calculate all combination possible keeping order of init fixed i.e,

Combinations = [[1,2],[2,1]]

let, theta = 0(degrees)

Calculate the angles between each combination vector and predicted_array's first two dimensions A1 = $\cos^{-1}$(dot product ([1,2],[1,2]) = 0 degrees, 0 is not greater than 0 which implies no update of theta.

A2 = $\cos^{-1}$(dot product ([2,1],[1,2]) = 37 degrees, i.e theta<A2<90, hence, theta = A2 = 37 degrees.

Update init as init = [2, 1].

Keeping the order of values in init as constant, add new dimension value from predicted_array and calculate all combinations again as in step2.

Init = [2, 1],

Combinations = [[3,2,1],[2,3,1],[2,1,3]],

theta = 37 degrees

A1 = $\cos^{-1}$(dot product ([3, 2, 1], [1, 2, 3]) = 44.41 and as theta<A1<90, hence, theta = A1 = 44.41 degrees.

A2 = $\cos^{-1}$(dot product ([2, 3, 1], [1, 2, 3]) = 38.2 and as theta>A2 and theta<90, hence, no update.

A3 = $\cos^{-1}$(dot product([2,1,3],[1,2,3]) = 21.7 and as theta>A3 and theta<90 ,
hence, no update.

Hence, init is updated to [3, 2, 1]

Finally, we can say the required vector = [3, 2, 1] which is at an angle 44.41 degrees to [1, 2, 3].
a 90-degree combination is not possible with [1,2,3], hence we got near a 90-degree combination. Another example, when the phase shift algorithm is applied on the vector [0, 0, 1], we get the vector [1, 0, 0] as shown in Figure 11.

With this algorithm, we can do the desired rotation of an array or vector with a complexity $O(n^2)$ which is a tremendous speed up as compared to the O(n!) permutation method.

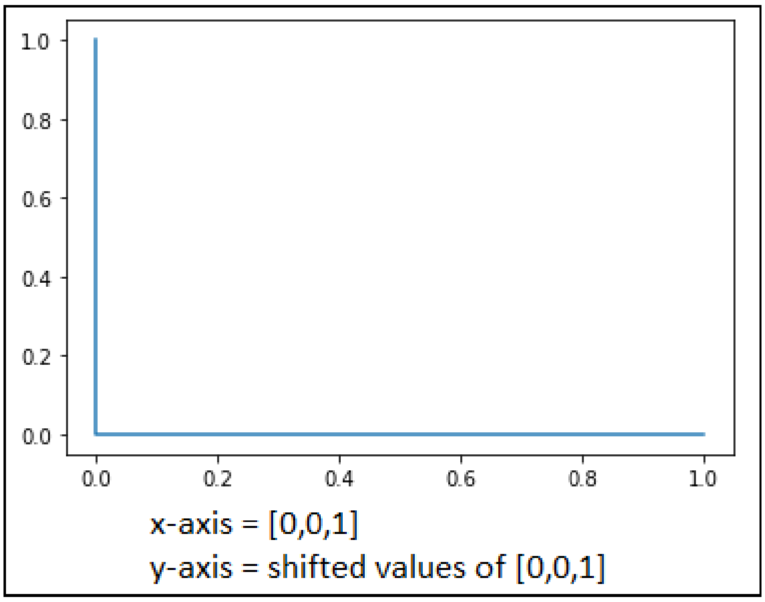


Figure 11. Graph representing a perfect shift of 90 degree for vector [0,0,1] to [1,0,0]

4. DISCUSSION

The following points are important results we got from this method:

The song and the background music of the song are mostly and approximately perpendicular to each other.

The ascending order arrangement of the music audio is mostly and approximately perpendicular to the original arrangement of the same music audio.

Gaussian distribution is used to get excellent compression of data and reduces data to (number of intervals/size of audio file array)*100% of the actual size of the data.

The phase shift algorithm rotates a given array to a given angle combination of the array if possible and if the given angle combination is not possible, it gives a near-angle combination of the array i.e an angle close to 90 degrees.

Gaussian trick enables an individual to store maximum data but at a cost of losing the sequence or arrangement of amplitudes over the time frame of the signal.

5. CONCLUSION

Generator algorithms are growing diverse as time passes incorporating better changes and many applications are also being developed on the same. The music field is one of the fields where generator applications can perform very well and do the need, by creating or generating good music and give good ideas to musicians and helping them put forward their best while composing the music. This application can serve as a resource to many in the music field and also help many lyricists and give them ideas in better and diverse ways. Generator applications can also be used in the literature field to understand how computers, if given a chance, would write, compose and speak, which is really exciting for many researchers to understand the complexities and develop better algorithms.

REFERENCES

[1] Dhariwal, Prafulla, et al. "Jukebox: A generative model for music." *arXiv preprint arXiv:2005.00341* (2020).

[2] Wang, Xin, et al. "A Vector Quantized Variational Autoencoder (VQ-VAE) Autoregressive Neural $ F_0 $ Model for Statistical Parametric Speech Synthesis." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2019): 157-170.

[3] Choi, Kristy, et al. "Encoding musical style with transformer autoencoders." International Conference on Machine Learning. PMLR, 2020.

[4] Chorowski, Jan, et al. "Attention-based models for speech recognition." *arXiv preprint arXiv:1506.07503* (2015).

[5] Gupta, Swati. "A regression modeling technique on data mining." *International Journal of Computer Applications* 116.9 (2015).

[6] Cochran, William T., et al. "What is the fast Fourier transform?." *Proceedings of the IEEE* 55.10 (1967): 1664-1674.

[7] Krithikadatta, Jogikalmat. "Normal distribution." *Journal of conservative dentistry: JCD* 17.1 (2014): 96.

[8] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

[9] Maind, Sonali B., and Priyanka Wankar. "Research paper on basic of artificial neural network." *International Journal on Recent and Innovation Trends in Computing and Communication* 2.1 (2014): 96-100.

[10] LeCun, Yann, and YoshuaBengio. "Convolutional networks for images, speech, and time series." *The handbook of brain theory and neural networks* 3361.10 (1995): 1995.

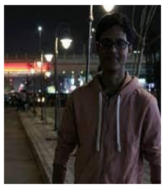
[11] Gupta, Lalit, Mark McAvoy, and James Phegley. "Classification of temporal sequences via prediction using the simple recurrent neural network." *Pattern Recognition* 33.10 (2000): 1759-1770.

[12] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

[13] Goodfellow, Ian, et al. *Deep learning*. Vol. 1. No. 2. Cambridge: MIT press, 2016.

AUTHORS

V. N. Aditya Datta Chivukula is currently an undergraduate student in the Department of Computer Science and Engineering at International Institute of Information Technology, India. His area of interest in Machine learning, Deep learning, etc.



Abhiram Reddy Cholleti is currently an undergraduate student in the Department of Electronics and Telecom Engineering at International Institute of Informational Technology, India. His area of interest in antenna design, IoT, etc.

Prof. Rakesh Chandra Balabantaray research interests lie in the areas of Intelligent Systems, Data Mining, Information Retrieval, and Natural Language Processing. He teaches Information Retrieval and Data Mining, Web Mining and Natural Language Processing. He has published over 100 research papers. He has copyrights for the software he has developed. He was also the principal investigator of a project on the Development of Cross Lingual Information Access sponsored by DIT, GOI.