THE SONG MACHINE: GENERATING MUSIC FROM AUDIO AND MIDI

Alessandro Cerioli
Aalborg University
acerio19@student.aau.dk

Carmen Muñoz
Aalborg University
cmunoz19@student.aau.dk

ABSTRACT

Audio generation has been addressed in many ways with the focus to reproduce hours of a music style or specific voice inter alia [1][2]. However, little has been done to actually resemble a song within its characteristic content structure. We can highlight some attempt [3] but it is a constraint song generation as it needs some input/guideline from a user. The Song Machine (TSM henceforth) tries to reproduce a whole song which sound resembles the one by a particular music band following two different approaches: audio and midi. It works on the song sections of the band song dataset to extract its features to build up a new song. Our study is unique differing from other researches as it focuses on song sections (chorus, bridge, intro...) rather than only the whole song as we want to reproduce a song with a certain structure elements that repeat over time.

1. INTRODUCTION

The main goal of this mini project is to attempt to generate a song that resembles a particular band using audio and midi dataset from this group. Two distinct methods are considered then, an audio approach and a midi approach so we can compare results at the end project. Audio approach makes use of autoencoders for audio generation and midi approach uses recursive neural network. Both work on song sections rather than just a full song.

2. BACKGROUND AND LITERATURE REVIEW

2.1 Audio Approach

Music production with machine learning has been addressed a few times. Using raw audio [2], spectrograms [4] or midi as inputs. For this purpose, NNs such as Wavenets [4], GANs [5], convolutional NNs [6] have been used, but an autoencoder generation approach for music is still an open field for study even greater when it comes to unconventional latent space feeding. In addition, music has been generated in "random" forms, without following a fixed structure that modern songs have. This mini project works on these issues.

2.2 Midi approach

In a MIDI context, even if the aim was the same as for the audio, the approach was quite different because of the dissimilar format of the information. As a matter of fact, in this case, it is not possible to work on the spectrogram of the song, but instead on the sequential structure of the notes in time. Consequently, it has been more advantageous the use of a recurrent neural network [7] instead of a convolutional neural network and autoencoder. It allows to predict new instances of the sequence, on the basis of the previous ones and this memory is necessary to reproduce the logic arrangement of a musical composition.
The use of neural network in this field represents a step forward regarding the artificial intelligence related to music composition. In fact, before the affirmation of neural networks as one of the most powerful formalisms of machine learning, the only very relevant mathematical tool available were Markov chains [8]. The research in this modality of automatic composition and generative music is hence a completely new field to explore.

3. DESIGN AND IMPLEMENTATION

Song section will be written abbreviated as SS. Notice SS**s** when referring to the plural. A SS is a specific part of the song (i.e.: chorus, intro, end, riff, pre-chorus, bridge…).

3.1 Audio Approach

In summary, TSM when it comes to audio takes all songs from a band, splits then into SSs, learn specifically from them its features and generates a new SS. Afterwards, the SSs are joined together to form a song with a particular structure.

3.1.1 Building the SS dataset

For this purpose, an excel document was designed listing some songs from the album "One" by the Beatles and the starting and ending seconds of each SS, each one

annotated by hand while a not-short listening of the album.

Band	Song	Part	Start(s)	Ending(s)	Difference(s)
theBeatles	LoveMeDo	i	0	13	13
theBeatles	LoveMeDo	c	14	26	12
theBeatles	LoveMeDo	r	27	34	7
theBeatles	LoveMeDo	c	35	48	13
theBeatles	LoveMeDo	r	49	56	7
theBeatles	LoveMeDo	va	57	68	11

Table 1. Some rows of the excel containing data from songs

To avoid outliers that bias profoundly AE learning and training, statistical measurements were taken such as the mean and standard variation for each SS.

Stadistics of songSections	mean	std
i	8	3200
v	7,639175258	3700
va	6	3360
p	7,05	3170
c	7,597222222	3120
b	10,4	3810
s	12	5.686
r	6	3.415
e	11,64285714	7570

Table 2. Means and stds of SSs

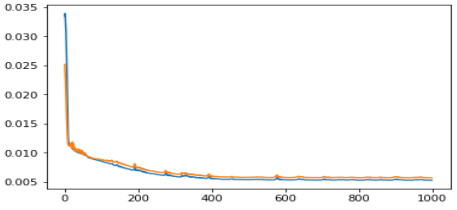
All SSs duration which do not lie in between *mean+-std* were discarded. For example, the song "Yesterday" which is at a slower tempo than the mode songs, like "Love me do" was rejected for the study. Consequently, a python program was designed to cut all this SSs automatically in pieces by reading the information from the excel document and discriminate them if it is considered an outlier. Afterwards, time-stretching (with pitch-shifting) post-processing was used to enlarge or decrease each SS duration to fit the mean SS duration. The milestone of all SSs fitting the mean duration is so as all inputs to the AE will contribute the whole-time range. If an input is shorter in time than other, it will stop contributing to the generated output when it finishes, as it is made of '0's. Finally, each SS is exported individually in different SSs folders while keeping a correct labelling for classification if of interest for future implementations (more than 300 files are generated). The used labelling is: *NameOfSong_orderOfAppearanceInDataset_SS.wav.* It may seem a tedious work to do and implement from the very beginning, but we strongly believe that if we want to reproduce a song which is divided and built with different SSs and no song is really similar to other in the whole structure, we need to work with SSs as a reference and not just the whole song as other studies do because in the use of GNN, the input will be strictly related with the output, and it is not of our interest to produce hearable music without a recognizable structure [4], that makes it actually not so pleasant to listen to as we do with normal songs that is our main goal.

3.1.2 From SS to spectrogram input of AE

Having cut and classified all SSs, a mel-spectrogram was computed for each SS wav after a mono conversion. Mel spectrogram was proposed rather than computing a bare spectrogram because of its approximation when it comes to audio feature extraction in relation to human auditory system response [9] as well as used in other GNN approaches [4][5][6]. The computed mel spectrograms have all same dimensions which can be modified by changing the tunable parameter "Mel spectrogram coefficients". At this point of the algorithm, all inputs (mel-spectrograms) have same dimension facilitating the feeding of the AE in contrast to other algorithms which work on raw audio [2][10] as input of the GNN. This approach is also due to the short time duration on inputs to study (SSs), which will not be worthy in algorithms using LSTM that rather study an entire song and memory about all parts (but without distinction) must be considered as important [10]. That is the reason why its outputs have not a exactly song format which are more like a mix of all parts of it. Afterwards, the matrix that represents the mel spectrogram is normalized (not to bias AE) and restructured into a vector. The number of elements of the vector is actually the dimension of the input layer of the AE.

3.1.3 AE design and training

A simple AE with 2 layers and Dense configuration is developed. Results from encoded original SSs at sampling rate 16.000Hz (sampling rate is a tunable parameter) are incredibly nice for such a bare structure, however, it introduces an annoying harmonic periodic



noise that self-repeats over the whole spectrogram. Used batch size is 256, loss function 'mse' and 'adam' optimizer. Loss validation function get constant around 500 epochs.

Figure 2. Loss validation function for 1000 epoch training

Repeating 1000 epochs more does not get better:

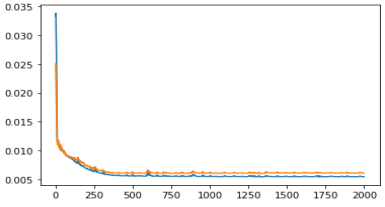


Figure 3. Loss validation function for 2000 epoch training. A convolutional neural network has also been applied to compare functionality; however, loss function gets a really worse result than basic AE.

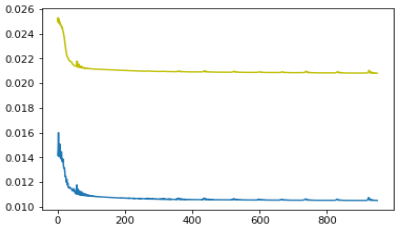


Figure 4. Loss validation function for 1000 epoch training for Convolutional NN. Train loss (yellow) and validation loss (blue).

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	[(None, 16, 41, 8)]	0
conv2d_18 (Conv2D)	(None, 16, 41, 8)	584
up_sampling2d_9 (UpSampling2	(None, 32, 82, 8)	0
conv2d_19 (Conv2D)	(None, 32, 82, 8)	584
up_sampling2d_10 (UpSampling	(None, 64, 164, 8)	0
conv2d_20 (Conv2D)	(None, 64, 164, 16)	1168
up_sampling2d_11 (UpSampling	(None, 128, 328, 16)	0
conv2d_21 (Conv2D)	(None, 128, 328, 1)	145

Figure 5. Summary of the ConvNN

3.1.4 AE to generate new songs

After training, a validation or test of the first 20% of the training set is used. If we encode one song and we decode the encoded song, the output is a "reconstruction" of the song, but actually we are not generating "new" data taking into account all the learning processes but rather the "memorized" learning to decode that song.

This is a problem as our goal is to generate a song that has a "mixture" of songs rather than repeating it with its encoded version. The way to generate new data taking into account more than one encoded SS is to combine encoded songs. Therefore, the latent space of the AE is fed with an encoded SS which has been built out of combining other encoded SSs.

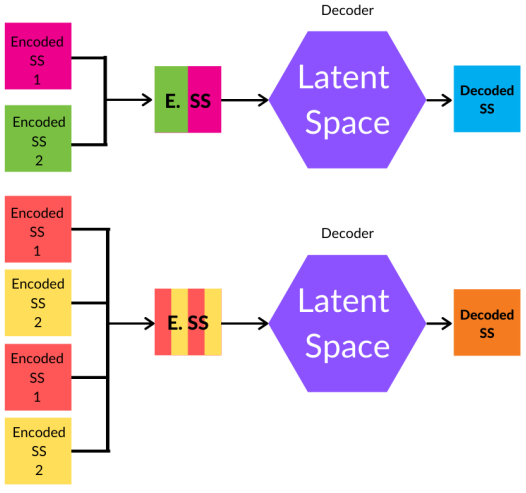


Figure 6. Graphical description of 2 possibilities (halves and fourths) of combining 2 encoded SS to feed latent space

3.1.5 Post processing

As previously mentioned, AE introduces a disturbing harmonic periodic noise. It is also could be due to the low-quality recording of the album in the '70's which has already some noticeable high background noise. A procedure to avoid partly this noise is to high pass filter the signal with fc=160Hz.

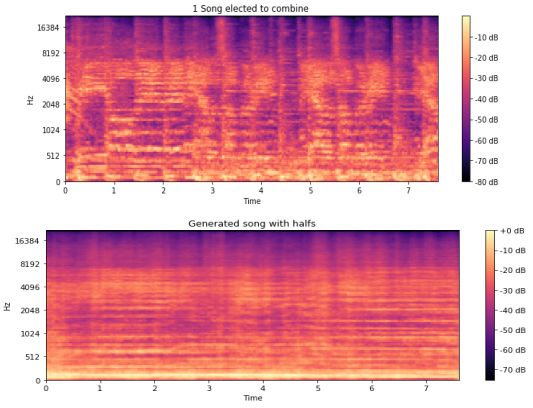


Figure 7. Mel spectrogram of a SS and of a generated SS without post-processing, notice high continuous noise at low frequencies.

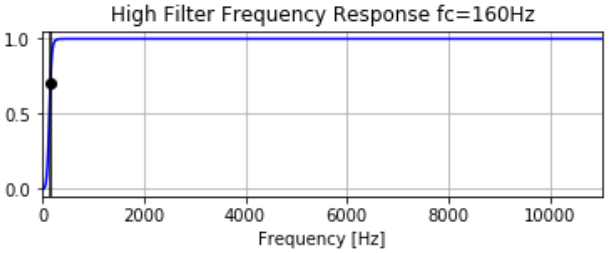


Figure 8. High pass filter used

Last but not less important, a stereo effect is implemented in python to pursue a better mix and perception of the audio signal. Finally, normalization is applied to recover the energy lost along the process.

3.1.6 Joining SS

After having ran TSM on the different SSs, we can join all of them following the structure we want to output. A general one used is the so-popular intro/verse/pre-chorus/chorus/verse/pre-chorus/chorus/bridge/chorus/end but the user can change the desired song output structure.

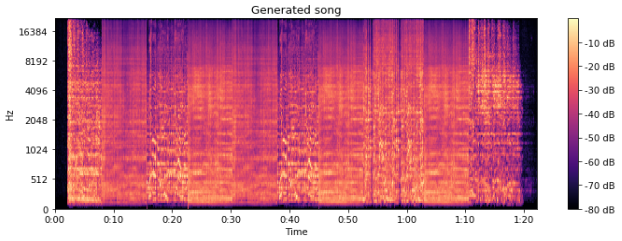


Figure 9. Mel Spectrogram of the outputed song. Notice spectral repetitions.

3.2 Midi approach

3.2.1 Dataset creation

The first step of the implementation consisted in creating a coherent dataset. In order to get it, it has been necessary a search for MIDI files of pieces by the author we wanted to simulate the style.

A set of MIDI files have been downloaded from MuseScore [11] website and, before proceeding, their coherence has been qualitatively checked by using Audacity software [12].

In particular for TSM trying to simulate the style of songs by The Beatles, the songs from the collection album "One" were used.

As explained in the introduction, the purpose was to maintain the structure of the song, so that the composition is coherent in his different parts. Hence**,** the pieces have been subdivided in ten sections in percentiles on the basis of the number of notes (for example, the first section will contain the first 10% of the notes of the pieces, see Figure 10). Also, it is worth considering that silences are considered notes. The chosen of the number ten was arbitrarily, because it was in our opinion a good compromise between the number of sections to subdivide the piece and the number of notes in a single section. Probably, it does not exist a universal suitable number, likely it depends on the number of notes in the piece.

On average, the songs contained 5676.909 notes (considering all the different instruments) and therefore 5676.909/10 = 567.6909 notes in a single section is a good number not to fragment too much the piece and not to lose its structure.

Since it is not possible to know a priori the optimal number of sections, it was important to leave in the code the possibility to decide the number of sections as a parameter, so that it will be possible to reuse such code in eventual future experiments.

All the notes were ordered in each section on the basis of the start time, considering both melody and musical accompaniment.

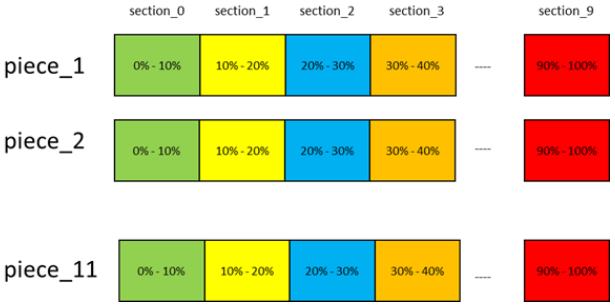


Figure 10. Songs division in sections (not all the songs from the album have been used, for dataset size and computational time matters)

After having obtained the sections, the next step was getting the chunks of every single section. Sequences of 31 notes were chosen. Also in this case, 31 has nothing special, but from a qualitative point of view, it maintains the style of the composer and at the same time does not reduce too much the number of sub sequences with which the datasets (one for each section) will be created.

Obviously, the longer is the sub sequence, the lower will be the number of the possible subsequences, according to the binomial coefficient equation (1):

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad (1)$$

This, as the number of sections, could be not so good in other contexts.

With Beatles songs subdivided in 10 sections chunked in sequences of 31 notes, the following result has been obtained: 4479 different subsequences for the first section, 6425 for the second one, 6615 for the third one, 6616 for the fourth one, 6624 for the fifth one, 6289 for the sixth one, 5726 for the seventh one, 6980 for the eighth one, 6050 for the ninth one, 3432 for the tenth one; for a complexive number of 59036 subsequences with which finally the datasets has been built.

To create the dataset, these features were considered for each note: pitch, duration (note_off – note_on events) and start_time (note_on event). Hence, the dataset will have 3 x 31 = 93 columns. Consequently, the size of the dataset will be 59036 x 93 (see Figure 11).

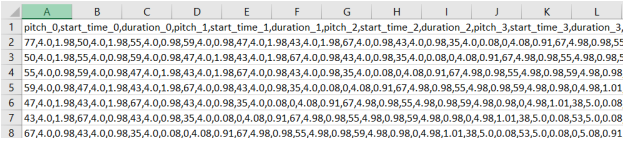


Figure 11. Example of a little portion of dataset

3.2.2 Recurrent Neural Networks training

Once the dataset was ready, the next step was feeding the neural network. In particular, we decided to use recurrent neural networks (long short-term memory neural network) because the model should have been trained to recognize sequences and to predict a new value given a new sequence in input.

For this purpose, a recurrent neural network for each

component of MIDI note (pitch, duration, start time) and for each section of the piece has been used. Since the number of components is three and the number of sections is ten, the total number of neural networks trained was thirty. In order to choose the section to train, different scripts have been created. Moreover, to train one of the three components of the MIDI note, we simply chose as target the last instance in the sequence of the component we wanted to predict and we dropped such column in the features. For example, to predict the pitch, the structure was:

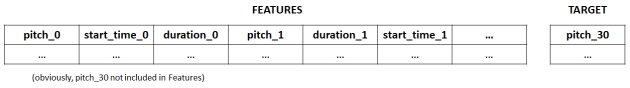
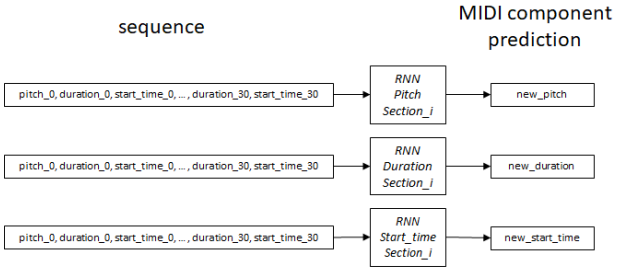


Figure 12. Features/Target structure to feed pitch prediction Neural Network

This process has been iterated for thirty times, or rather to train each neural network (see Figure 13)..



$$0 \leq i < 30$$

*F*igure 13. Recurrent Neural Network prediction mechanism

3.2.3 Song reconstruction

Once all the neural networks were trained, the final step was creating the new piece. In order to create it, it has been necessary a main program. It takes an input text file containing all the information regarding a sequence of 31 notes (so 93 numbers, considering pitch, start time and duration) and a sequence of numbers representing the number of notes per each section. Once the data has been read, for each note in each section a new pitch, a new start time and a new duration are predicted (on the basis of the current sequence), then the first note of the sequence is discarded and the new one is appended at the end. Then, the procedure is iterated until the end of the number of notes. Also, each predicted note is appended to the list of the notes of the composition (see Figure 14).

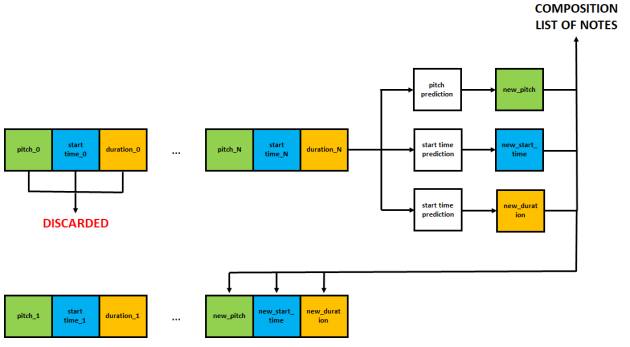


Figure 14. Block diagram of the general structure of the main program to re-create the single sections.

4. RESULTS

Result files, audios, codes and midis are in this link.

4.1 Audio approach

Depending on the chosen SSs to feed the latent space, a more or less acceptable outcome is obtained. By increasing the sampling rate to 44100 Hz results get better in term of periodic noise. Also, the use of post-processing makes a difference when hearing results. Varying the latent space by a compression factor makes not much difference, getting really similar outputs when using 5 or 20 as compression factor. Further than cFact= 100, it gets each time more noise. Mixing fourths instead of halves leads to a better intelligible music. Mixing four different SSs to produce a SS lead to mostly noise. Using a convolutional NN although having a little more loss error, outputs disturbing periodic noise. Applying one LSTM layer to basic AE outputs lots of noise, taking a huge time for training and high loss (4 hours-20 epoch-loss=0.02).

4.2 Midi approach

It is worth considering that the components of the MIDI note have been used(pitch, duration and start time) are comprised in a range that is clearly suited to them, without evident outliers. Moreover, it is possible to hear the two different streams of melody and accompaniment. The result can significantly change considering the chosen parameters and the dataset. As a matter of fact, another experiment with the same parameters (length of the sequences, number of sections and so forth) but different dataset was conducted: it consisted in using nocturnes by Chopin instead of The Beatles songs. It is possible to hear how different datasets bring to different results and how it is necessary to find a new balance of the parameters, considering different artists.

5. DISCUSSION

5.1 Audio approach

The generated song depends radically on the encoded SSs chosen to build the SS. A bad combination of encoded SSs will lead to awful, noisy results. So, particular combinations must be considered when feeding the latent space. As more similar the encoded SSs are, the more “hearable” result will be outputted, but it is not always the rule. Therefore, based on what previously said, combining more than 2 different SSs is riskier to obtain a nice output than just two. Moreover, it usually adds more undesired noise. Additionally, the low quality initially chosen data may have led to the general noise at the generated data. As our AE design does not take into account the contribution of SSs among each other (the

whole entire song), a noticeable change in the form or "timbre" strikes when transiting from each SS to another.

5.2 Midi approach

The choice of using a recurrent neural network to generate music is probably, at the moment, the most suited. Nevertheless, in order to get good results, the number of factors to consider are high. For example, deciding a greater or a smaller number of sections affects significatively the result, because it affects the number of sequences for the single neural network training (for each section) and the measure of the fragmentation of the piece.
Also, the number of notes for each sequence is a fundamental parameter, because they should be long enough to preserve the composer style, but not too long so that the dataset will contain a greater number of instances.
Finally, it is worth to consider a high number of pieces by the same author with a similar style, to get a higher number of sequences.

6. CONCLUSION

6.1 Audio approach

Simple AE approach can lead to impressive results comparing the decoded encoded-input with the original input, but in terms of generating "new" data as a mixture of the inputs by combining encoded pieces to feed the latent space, it may lead to randomly good or bad results. It is worth to highlight fair good results even when using a compression factor of 50 for reducing latent space. Although, never in literature has been latent space contemplated to receive such input combination, and this method, used in the wise manner and with other data type as inputs like images and considering more statistical data, may be a cheap computation way to explore a different way of generating new data out of the chosen combination. Other approaches like LSTM or convolutional NN do not get acceptable results with so low training as basic AE, however, with far more epochs to train, they could get even better on behalf of computational time and complexity.

6.2 Midi approach

Approaching a problem of music composition by considering in the same sequences more instruments (and hence more musical streams) can be a good solution to obtain a coherent accompagnement for the melody. Also, in this way, more sequences could be available for neural network training. Moreover, since MIDI is a data structure based essentially on temporal parameters such as note_on and note_off events, a recursive neural network is likely the best solution to try to predict all the future parameters on the basis of the previous ones. Finally, another fundamental point, is maintaining joint into the same sequences the different MIDI parameters (start_time, duration, pitch) because they are not independent, but, conversely, they affect each other for the future forecasts.

7. FUTURE WORK

7.1 Audio approach

As this AE idea has never been explored in general, changing the parameters of the network (optimizer, loss function, neurons...) may lead to better results. Besides, a NN can be computed for post/pre-processing to denoise low-quality data such as ours. Lastly, another AE could be designed for understanding the "form" of the whole song and in combination of the AE trained on the SSs, generate a more coherent song among its parts.

7.2 Midi approach

To improve the current result in the future with this approach, it will be useful to reconsider the parameters of dataset creation. Or rather, a greater number of songs could be used, increasing the dimension of the dataset. Also, a different number of sections and a different length of subsequences could be considered, because they affected a lot the way the piece is structured and the fidelity with which the style of the composer will be conserved in the predictions. Also, a greater number of epochs for the training will be able to drastically improve the quality of the MIDI components forecasts. Furthermore, in order to maintain the higher level possible of coherence regarding the pitches, an improvement for next experiments could be transposing all the songs in the same tonality. This is also an advantage for the neural network training because in such a way it has a smaller range of pitches from which to choose, or rather a smaller number of possible outputs.

8. REFERENCES

[1] Mehri, Soroush, et al. "SampleRNN: An unconditional end-to-end neural audio generation model." *arXiv preprint arXiv:1612.07837* (2016).
[2] Carr, C. J., and Zack Zukowski. "Curating Generative Raw Audio Music with DOME." *IUI Workshops*. 2019.
[3] Liao Q., Yang N., Luan J., Wei F.: *AUTOMATIC SONG GENERATION.* INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT CORPORATION TREATY (PCT)
[4] Shen, J.; Pang, R.; Weiss, R. J.; Schuster, M.; Jaitly, N.; Yang, Z.; Chen, Z.; Zhang, Y.; Wang, Y.; Skerry-Ryan, R.; Saurous, R. A.; Agiomyrgiannakis, Y.; and Wu, Y. 2017. *Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions*. eprint arXiv:1712.05884.
[5] Lauri Juvela, Bajibabu Bollepalli, Junichi Yamagishi, Paavo Alku. *GELP: GAN-Excited Linear Prediction for Speech Synthesis from Mel-spectrogram*

[6] Federico Colangelo, Federica B., Alessandro N., Marco C. *CONVOLUTIONAL RECURRENT NEURAL NETWORK FOR AUDIO EVENTS CLASSIFICATION.* Technical Report

[7] Chen, C-CJ; Miikkulainen, Risto; IJCNN'01. "International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)"; pp 2241--2246; 2001

[8] Dubnov, Shlomo and Assayag, Gerard and Lartillot, Olivier and Bejerano, Gill; *"Using machine-learning methods for musical style modeling*"; pp 73-80; 2003

[9]https://en.wikipedia.org/wiki/Mel-frequency_cepstrum

[10] Carr, C. J., and Zack Zukowski. "Generating Albums with SampleRNN to Imitate Metal, Rock, and Punk Bands." *arXiv preprint arXiv:1811.06633* (2018).

[11] MuseScore; https://musescore.com/dashboard

[12] Audacity https://www.audacityteam.org/