2024 Brazilian Technology Symposium

Generative Solutions for Musical Composition

Lucas S. Scocca e Magno T. M. Silva
Electronic Systems Engineering Department
Escola Politénica, University of São Paulo
São Paulo-SP, Brazil
{lucas.scocca, magno.silva}@usp.br

***Abstract*— This paper addresses the generation of classical music using generative solutions with machine learning techniques. We propose two different music generators: one based on a generative adversarial network (GAN) that treats music as a matrix and another based on natural language processing (NLP) that seeks to capture temporal dependencies in music by using a long short-term memory (LSTM) recurrent network. Preliminary results indicate that NLP-LSTM-based technique produces qualitatively superior music, showing the potential of this approach in classical music generation.**

***Keywords*— machine learning; neural networks; generative adversarial network; natural language processing; music-generating techniques.**

I. Introduction

The growing interest and importance of generative artificial intelligence (AI) have been evidenced by the proliferation of tools that explore the autonomous and creative creation of images and music [1]–[5]. In the field of music, the motivation behind this lies in the perception that good instrumental music generators can have significant commercial applications, helping both independent creators and companies to obtain musical compositions more accessible and in a efficient manner [5].

Generative adversarial networks (GANs) [4], [6] and natural language processing (NLP) [7], [8] techniques have emerged as promising approaches in generating creative content. GANs with their generator/discriminator model have transformed how we perceive the creation of music and digital art by enabling the automated production of original and captivating works [1]. On the other hand, long short-term memory (LSTM) networks [9], [10], widely used in NLP, have shown great potential in capturing complex temporal dependencies, making them suitable for generating coherent musical sequences.

The literature review covers various music generation experiences with GANs by focusing on the use of musical instrument digital interface (MIDI) databases for training. An example is MP3net [11], which uses a 2D convolutional GAN trained with the MAESTRO dataset [12], generating high-quality audio through MP3/Vorbis compression techniques. Another approach is the conditional GAN for electronic music genres [13], which introduces genre ambiguity in rhythmic pattern generation, encouraging musical originality. Additionally, the LSTM GAN for melodic music generation [14] explores the unique ability of LSTM to capture long-term correlations in musical data, yielding results considered pleasant and coherent in human evaluations. A relevant example of music generation with NLP can be found in [2], in which an LSTM network is used to generate polyphonic music. The proposed model is trained using a dataset of MIDI files, focusing on the ability of LSTM to capture temporal dependencies and generate coherent sequences of notes. The results of [2] show that LSTM-based models can successfully generate music that retains the stylistic elements of the training data. Recently, a trend towards generative AI with diffusion probabilistic models (see, e.g., [15] and its references) has been observed. This is the case of Suno AI (`https://www.suno.ai`), which is a digital platform that can generate personalized music based on text and make music composition simple, allowing everyone the experience of creating music [5].

In this paper, we propose two classical music generators: one based on GAN that treats music as a matrix and another based on NLP-LSTM that seeks to capture temporal dependencies in music. In order to explore the potential of GAN and NLP-LSTM techniques, musical compositions are treated as a matrix and in a sequential manner. As dataset, we consider Johann Sebastian Bach masterpieces for piano. The synthetic generated musical compositions were classified by a network trained with three different genres of "real" music: Bach, Mozart, and music generated by AI. The correct classification of the Bach synthetic musical compositions indicates that our models can capture somehow the Bach style.

The paper is organized as follows. Section II presents the theoretical foundations of GAN, NLP, and LSTM. Section III describes the methodology used by including the dataset, data format, preprocessing, and post-processing as well as the architectures of the proposed generative models. Section IV presents the results and discussion by comparing GAN and NLP-LSTM-based approaches. Section V closes the paper with conclusions and future works.

II. Revisiting machine learning models

In this section, we describe the main foundations of the machine leaning techniques used in our models. The section is divided in two subsections dedicated respectively to GAN and NLP-LSTM.

This work was supported by CAPES under Grant 88887.717846/2022-00 and Finance Code 001 and by CNPq under Grants 303826/2022-3 and 404081/2023-1.

A. GAN

GANs are machine learning models consisting of two competing neural networks: the generator and the discriminator [4], [6]. The generator creates new samples from noise that resemble the training dataset, while the discriminator assesses whether a sample is real (from the dataset) or fake (generated by the generator). This competitive process, illustrated in Figure 1, leads to continuous improvement of both models, resulting in increasingly realistic data generation [4], [6]. The cost function that guides this process is given by $\min_G \max_D V(D,G)$ [4], [6], where

$$V(D,G) = E_{p_{\text{data}}(x)}\left[\log D(x)\right] + E_{p_z(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

represents how well the discriminator D can distinguish real data x with distribution $p_{\text{data}}(x)$ from data generated by the generator $G(z)$, which receives noise with distribution $p_z(z)$, and $E[\cdot]$ represents the mathematical expectation operator. The discriminator's goal is to maximize $V(D,G)$, while the generator's goal is to minimize it. As shown in [6], the generator G is optimal when the discriminator D is maximally confused and cannot distinguish real samples from the fake ones [4].

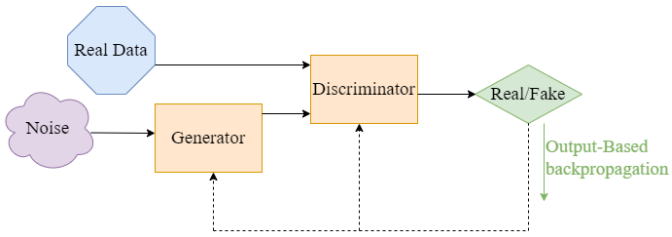

Fig. 1: Simplified scheme of GAN.

In addition to this GAN model, variants have been proposed to improve data generation in different contexts. For example, the deep convolutional GAN (DCGAN) was developed from an extensive exploration of convolutional neural network (CNN) architectures [16], [17]. In DCGAN, deep convolutions are able to capture spatial features in generated samples, being widely applied in tasks such as image generation. GANs have been also implemented with LSTM networks in order to be effective in capturing temporal dependencies in data series (see, e.g., [18]). These variants demonstrate the flexibility of GANs, allowing adaptations for different types of data and specific applications.

B. NLP-LSTM

NLP is a subfield of AI that focuses on the interaction between computers and natural human languages [7], [8]. The goal of NLP is to enable machines to understand, interpret, and generate language naturally for humans [7]. NLP techniques are widely used in a variety of applications, including machine translation, sentiment analysis, text generation, and speech recognition. One of the most important approaches in NLP is the use of LSTM, described next.

LSTM was proposed to solve the vanishing gradient problem of classical recurrent neural networks (RNNs) [9], [10]. The idea of this model is to create paths through time that have derivatives that neither vanish nor explode. They incorporate memory units and flow control mechanisms that allow them to preserve important information over long periods of time and regulate the flow of gradients during training. This enables them to model long-term dependencies in sequences without the gradient instability present in classical RNNs. As shown in Figure 2, a typical LSTM unit consists of a cell state, an input gate, an output gate, and a forget gate. In the figure, these components are represented respectively by $\mathbf{C}_t$, $\mathbf{I}_t$, $\mathbf{O}_t$, and $\mathbf{F}_t$. The cell maintains values over arbitrary time intervals, and the three gates control the flow of information into and out of the cell. The forget gates determine which information should be discarded from the previous state, assigning a value between 0 and 1 to the previous state compared to a current input. A value of 1 means preserving the information, while a value of 0 means discarding it. The input gates decide which parts of the new information should be stored in the current state, using a system similar to that of the forget gates. The output gates control which parts of the information in the current state should be produced, assigning values between 0 and 1 to the information, considering both the previous and current states. The selective production of relevant information from the current state allows the LSTM network to maintain useful long-term dependencies to make predictions in both current and future time steps. LSTM networks are generally composed of multiple cells like the one in Figure 2 connected together. The literature contains some variants of LSTM, as is the case of bidirectional LSTM (BiLSTM) [19]. A BiLSTM consists of two LSTMs: one that takes the input in a forward direction, and the another in a backward direction. This effectively increase the amount of information available to the network, improving the context available to the model. The ability of LSTM (or BiLSTM) of capturing long-term dependencies in data sequences is fundamental for tasks developed by NLP as language modeling, for example.

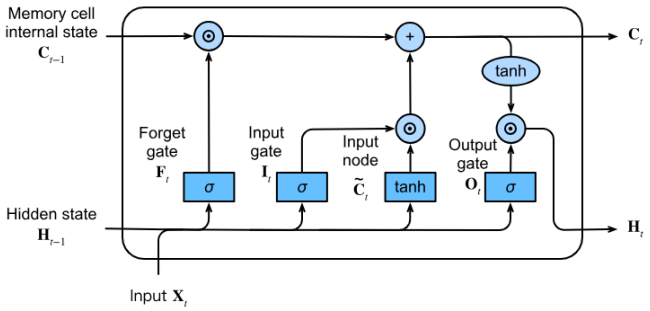

Fig. 2: Scheme of a typical LSTM network unit [20].

III. Methodology

In this section, the methodology used in this paper is detailed by including the dataset, data format, preprocessing, and post-processing as well as the proposed generative models.

A. Dataset and data format

To build the dataset, we use 25 piano scores of musical masterpieces by Johann Sebastian Bach in .MusicXML format, available at `http://www.mscorelib.`

com/actree/Bach/. To increase the dataset, each song was transposed (key change) with 12 semitones above and 12 semitones below, which leads to a dataset with 625 examples.

We use a simplified matrix representation for a music based on note structs. It offers a structured approach to capturing essential musical features, such as frequency, start time, duration, and instrument. This representation is based on the .MusicXML format, which facilitates conversion between formats and mathematical manipulation of notes. Table I presents a relationship between note names, notes in the simplified matrix representation format, and their actual frequencies in hertz. The simplified matrix format is used as input and output for the GAN and is exemplified in Figure 3.

This note representation is simpler than stating the frequency of the note itself, since in music theory notes are separated by semitones. Thus, a semitone difference[1] is equal to one in the representation for notes used. The mathematical relationship between two notes a semitone apart is $1/\sqrt[12]{2}$. Thus, if in the proposed format a note has the number assigned to the frequency of x, the relationship $\frac{x}{x+1}$ would be $1/\sqrt[12]{2}$ if we considered the real frequency. So, $x+12$ have twice the frequency of x, which is consistent with the idea of an octave, since a note one octave higher (or 12 semitones higher) has the same name, and has twice the frequency.

TABLE I: Relationship between note names, notes in the simplified matrix representation format (NS), and their actual frequencies (f) in hertz.

Note	C	C #	D	D #	E	F
NS	12	13	14	15	16	17
f (Hz)	16.35	17.32	18.35	19.45	20.60	21.83
Note	F #	G	G #	A	A #	B
NS	18	19	20	21	22	23
f (Hz)	23.12	24.50	25.96	27.50	29.14	30.87
Note	C	C #	D	D #	E	F
NS	24	25	26	27	28	29
f (Hz)	32.70	34.65	36.71	38.89	41.20	43.65
Note	F #	G	G #	A	A #	B
NS	30	31	32	33	34	35
f (Hz)	46.25	49.00	51.91	55.00	58.27	61.74

```
60 36 0 0 0 0 0 0
60 36 0 0 0 0 0 0
60 36 0 0 0 0 0 0
60 36 0 0 0 0 0 0
51 43 0 0 0 0 0 0
51 43 0 0 0 0 0 0
51 43 0 0 0 0 0 0
51 43 0 0 0 0 0 0
50 41 0 0 0 0 0 0
50 41 0 0 0 0 0 0
```

Fig. 3: Example of a Bach piece represented in the simplified matrix format.

In this format, each row represents the minimum time instance. Different notes played simultaneously are represented in the same row in different columns. The format supports up to eight notes played simultaneously and zero indicates that no note is being played. A row with only zeros indicates a pause. If two notes in the same column in consecutive rows are the same, it means that the note has a longer duration. If the same note is played repeatedly, it must be expressed in a different column in the following row, since a repeated note in the same column would result in a longer duration rather than the note played again. The example of Figure 3 follows the pattern from Table I. In this example, there are six columns containing only zeros, which indicates that only two notes are played simultaneously. The four first rows are equal to 60 36 0 0 0 0 0 0, which indicates that notes 60 and 36 played simultaneously has duration four times longer than the minimum. Note 60 has frequency $46.25 \times 2^{30/12} = 261.63$ Hz and note 36 has frequency $32.70 \times 2 = 65.40$ Hz and both represent note C. Figure 4 shows a scheme of how the music files are converted and manipulated.

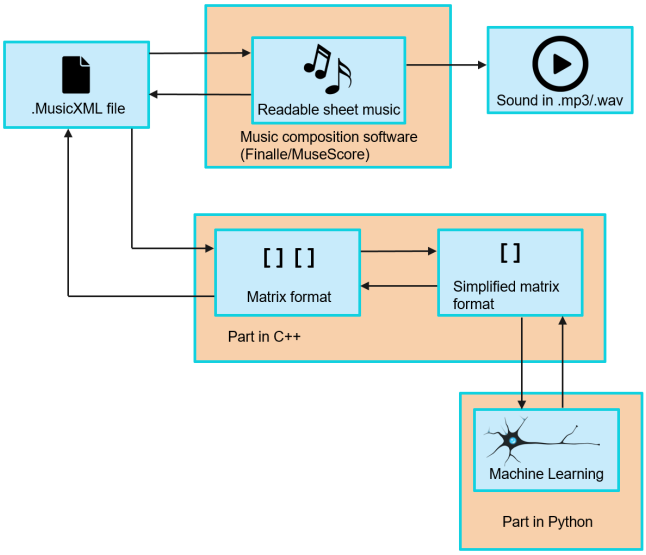

Fig. 4: Scheme of how the music files are converted and manipulated.

The same format and database are used in the NLP-LSTM approach. In this case, each row is treated as a token with the purpose of predicting the next set of notes in the sequence.

B. Data pre-processing and post-processing for GAN

Besides the aforementioned simplified matrix, no other pre-processing is done to generate music with GAN. In the post-processing, the matrices obtained at the output of the GAN are converted to integers (we use the Python function `astype(int)`). Arbitrarily, values less than or equal to 5 are approximated to 0. We should notice that the lowest notes in the music with a 12-semitone downward transposition[2] have values considerably greater than 5. Therefore, low values occur when the GAN tries to generate something intermediate between a pause and a note. An example of a generated music segment before (left) and after (right) post-processing can be seen in Figure 5.

[1]A semitone difference is the "smallest unit" of frequency in an instrument, like the difference of a fret on a guitar neck.

[2]This means the notes with the lowest number in the matrix representation minus 12.

```
52 30 0 0 0 0 -1 -1          52 30 0 0 0 0 0 0
52 25 0 1 0 -1 -1 0          52 25 0 0 0 0 0 0
50 34 0 0 1 -2 0 -1          50 34 0 0 0 0 0 0
55 29 0 0 0 0 0 -1           55 29 0 0 0 0 0 0
```

Fig. 5: Example of a generated music segment before (left) and after (right) post-processing in the simplified matrix format.

Furthermore, a post-processing to correct out-of-pitch of notes of the generated songs is taken into account, as explained next. We start by counting the number of each note. For example, suppose that a song contains 120 C, 111 C #, 100 D, and 90 G. Then, the total number is compared with all major scales, and the percentage of notes outside and inside the key is computed. Let us consider the C major scale, which normally has the notes C, D, E, F, G, A, and B, with no sharp or flat. In the example, we have $120 + 100 + 90 = 310$ notes within the key of C major and 111 notes outside the key of C. The scale that covers the most notes in the song is determined as the tonal scale of the song. Then, all notes outside the song's scale are corrected to the closest note within the song's scale. This post-processing makes the songs more pleasant to listen to.

C. Data pre-processing and post-processing for NLP-LSTM

Tokenization involves inserting a character, the letter 'a' in this case, between the spaces in the rows. For example, the rows

```
58 53 49 0 0 0 0 0
81 67 64 0 0 0 0 0
41 27 0 0 0 0 0 0
```

become

```
58a53a49a0a0a0a0a0
81a67a64a0a0a0a0a0
41a27a0a0a0a0a0a0
```

after the tokenization. The tokens are mapped using embedding vectors, so that the network can perceive the vector distance between words.

In the post-processing, the reverse process is performed by replacing 'a' characters with a space. Thus, the sequence of tokens is transformed into the simplified matrix format. Finally, the simplified matrix format is converted to .MusicXML format, which can be converted to .MIDI or .mp4 by various composition softwares.

The post-processing that corrects out-of-pitch notes used in songs generated with GAN was not used for songs generated with the NLP-LSTM model as they were all already within the song's key, indicating that the model used for generating music with NLP can better perceive which notes are in tune and out of tune.

D. Model architectures

We propose two GAN (G1 and G2) and one NLP-LSTM models for generating music. Model G1 is a conventional GAN with multilayer perceptron (MLP) networks [16] for both the generator and the discriminator, whose architecture is specified in Tables II and III. For the sake of compactness, in all tables presented in this section we use N for the number of filters or neurons per layer, P for pooling, D for dropout, LReLU(ℓ) for Leaky ReLU with parameter ℓ, and tanh for hyperbolic tangent.

Model G2 is a DCGAN with the hyperparameters shown in Tables IV and V.

TABLE II: Model G1 – Parameters for MLP-GAN discriminator.

Layer	N	Type	D	Activation
1 (in)	2^8	Dense	0.3	LReLU(0.2)
2	2^7	Dense	0.3	LReLU(0.2)
3	2^7	Dense	0.3	LReLU(0.2)
4 (out)	1	Dense	–	Sigmoid

TABLE III: Model G1 – Parameters for MLP-GAN generator.

Layer	N	Type	Activation
1 (in)	2^7	Dense	LReLU(0.2)
2	2^8	Dense	LReLU(0.2)
3 (out)	2^{15}	Dense	tanh

TABLE IV: Model G2 – Parameters of the DCGAN discriminator.

Layer	N	Type	P	Stride	D	Activation
1 (in)	16	Conv2D	(5,5)	(2,2)	0.3	LReLU(0.3)
2	32	Conv2D	(5,5)	(2,2)	0.3	LReLU(0.3)
2	64	Conv2D	(5,5)	(2,2)	0.3	LReLU(0.3)
4 (out)	1	Dense	–	–	–	Sigmoid

TABLE V: Model G2 – Parameters of the DCGAN generator.

Layer	N	Type	P	S	Activation
1 (in)	2^{11}	Dense	–	–	LReLU(0.3)
2	2^6	Conv2DTranspose	(5,5)	(2,2)	LReLU(0.3)
2	2^5	Conv2DTranspose	(5,5)	(2,2)	LReLU(0.3)
4	2^4	Conv2DTranspose	(5,5)	(2,2)	LReLU(0.3)
5 (out)	1	Conv2DTranspose	(5,5)	(1,1)	tanh

For music generation with NLP-LSTM, the hyperparameters includes an embedding layer with a dimension of 100, a BiLSTM layer with a dimension of 1000, and a dense layer with softmax activation. During the generation of new music, a logic was implemented to randomly select the top model predictions to introduce variability in the generated compositions.

All models were trained using the cross-entropy loss function and the Adam optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$, and batch size of 25 [16], [21]. The learning rate was set as $\eta = 2 \times 10^{-4}$ for Model G1, $\eta = 10^{-4}$ for Model G2, and $\eta = 10^{-2}$ for LSTM-NLP. The samples of music of the training dataset were adjusted to have the same duration. Shorter pieces were padded with rows of zeros at the end, which does not affect how the model interprets the first rows of the music. Each training sample consists in a song of the dataset, which is converted to a matrix of 4096×8 with the format of Figure 3. This matrix is considered as input for the discriminator of DCGAN (Model G2), it is transformed to a vector for the discriminator of MLP-GAN (Model G1), and tokenized for the LSTM-NLP. We used 500 out of 625 songs of

the dataset for training. Furthermore, noise of dimension 100 is considered as the input of G1 and G2 generators, whose output has the same dimension of each training sample, i.e., 4096×8.

IV. Results and Discussion

In this section, we present the results for three songs generated, named Music A, B, and C. These and other generated songs can be heard in a repository, available at `https://sites.google.com/usp.br/sitedolucas/`.

The MLP-GAN of Model G1 was trained during 500 epochs to generate Music A. Figure 6 shows the generator and the discriminator loss curves along the epochs for this case. We observe that these curves do not cross, which means that the discriminator and generator did not work one against the other as expected.

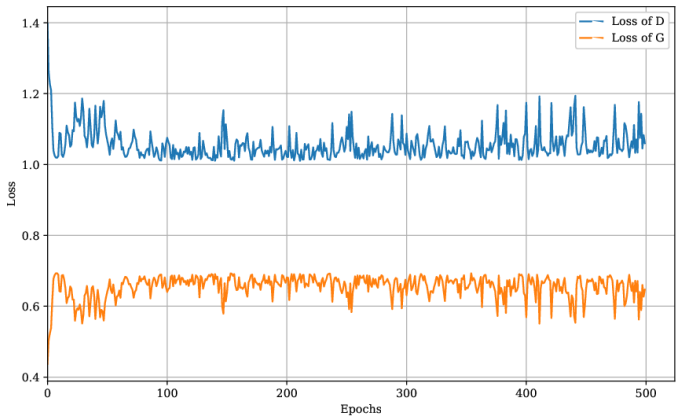

Fig. 6: The generator (G) and the discriminator (D) loss curves along the epochs for Model G1 that generated Music A.

The pattern of Bach's songs was more noticeable in Music B, generated with DCGAN of Model G2 which was trained during 400 epochs. Figure 7 shows the generator and the discriminator loss curves along the epochs for this case. Different from Figure 6, here we can observe that the loss curves intersect each other which means that the discriminator and generator do work one against the other as expected in a GAN. Indeed, Music B presents a more pleasant sound to listen to in comparison with Music A.

Music C was generated with NLP-LSTM, trained per 150 epochs. During the generation of this song, a logic was implemented to randomly select the top model predictions to introduce variability in the generated compositions. To measure the performance of this model, the accuracy curve was measured along the epochs and is shown in Figure 8. This metric in the context of an NLP token generation model reflects the proportion of correct predictions made by the model during training. Specifically, during each epoch, the model attempts to predict the next token in a sequence based on the preceding tokens. The accuracy is calculated by comparing the predicted token with the actual token in the training data. As the model iteratively adjusts its weights through the backpropagation algorithm, the predictions become more accurate, which is reflected in an increasing accuracy score

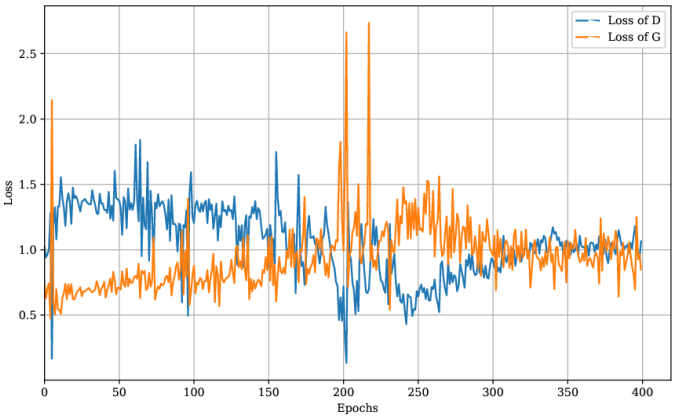

Fig. 7: The generator (G) and the discriminator (D) loss curves along the epochs for Model G2 that generated Music B.

over the epochs. The use of the accuracy metric is particularly relevant for this task because it provides a quantifiable measure of how well the model is learning to predict the next token in a sequence. The upward trend in the accuracy graph indicates that the model is improving its ability to generate sequences that closely resemble the patterns in the training data, which is a sign of the model is generating coherent and realistic music sequences. This behavior can be observed in Figure 8, since the accuracy of the proposed NLP-LSTM model achieves one after 120 epochs, which indicates that Music C is coherent. Comparing this song with those generated with the proposed GAN models, we can observe that it is it presents a more pleasant sound to listen to.

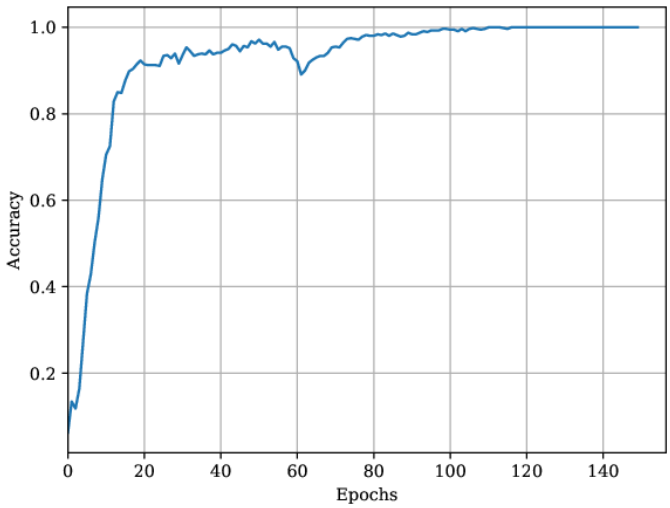

Fig. 8: Accuracy graph of the NLP model over the epochs.

In addition to the subjective evaluation, a discriminator was also carried out to check whether the generated songs actually resembled Bach's to a certain extent. This discriminator is an MLP network, whose parameters are shown in Table VI. It was trained to classify three different styles of music: Bach, Mozart, and songs generated by AI in a deterministic way. In the dataset, we consider 125 out of 625 songs of the aforementioned Bach dataset that were not

used for generating Music A, B, and C. The dataset was also formed by 125 musical masterpieces of Mozart, available at http://www.mscorelib.com/actree/Mozart/ and 125 songs generated by AI from the YACY software, available at https://www.youtube.com/@yacyai. 80% of the dataset was used for the training and 20% for the test of the discriminator.

TABLE VI: Parameters of the MLP discriminator used to evaluate the generates songs.

Layer	N	Type	Activation
1 (in)	512	Dense	ReLU
2	256	Dense	ReLU
3 (out)	3	Dense	Softmax

The trained discriminator achieved an accuracy equal to one when tested with Music A, B, and C. However, the output of the softmax function of the discriminator achieved values closer to 1 or 0 for Music C, when compared to Music A or B. This confirms that Music C, generated with NLP-LSTM, is closer to Bach's style than Music A and Music B, generated with GAN.

The training was done using a T4 GPU as a hardware accelerator. The training time for Music A was 79.33 seconds, for Music B, 757.55 seconds, and for Music C, 486.78 seconds. Since our preliminary results indicate that Music C is better than B and the latter is better than A, NLP-LSTM seems to be a suitable model for music generator than those based on GAN.

V. Conclusions

In this paper, we proposed three generative models for generate music. Different from the most of the solutions in the literature, we used a simplified matrix representation for music, based on notes and structs. The generated songs were evaluated with a discriminator that confirms what could be observed subjectively: the music generated with NLP-LSTM is more closely resembled real music style when compared to those generated with GAN models. In future works, we intend to compare our models to solutions proposed in the literature in terms of mean opinion score.

References

[1] S. Shahriar, "GAN computers generate arts? A survey on visual arts, music, and literary text generation using generative adversarial network," *Displays*, vol. 73, pp. 102237, 2022.

[2] J.-P. Briot, "From artificial neural networks to deep learning for music generation: history, concepts and trends," *Neural Comput. Appl.*, vol. 33, no. 1, pp. 39–65, jan 2021.

[3] H. Zhang, L. Xie, and K. Qi, "Implement music generation with GAN: A systematic review," in *International Conference on Computer Engineering and Application (ICCEA)*, 2021, pp. 352–355.

[4] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, pp. 53–65, 2018.

[5] J. Yu, S. Wu, G. Lu, Z. Li, and K. Zhang, "Suno: potential, prospects, and trends," *Frontiers of Information Technology & Electronic Engineering*, vol. 25, pp. 1025–1030, 2024.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Advances Neural Information Processing Systems Conference*, 2014, pp. 2672–2680.

[7] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing*, MIT Press, 1999.

[8] K. R. Chowdhary, *Natural Language Processing*, pp. 603–649, Springer India, New Delhi, 2020.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] A. Graves, *Long Short-Term Memory*, pp. 37–45, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[11] K. Van Den Broek, "MP3net: coherent, minute-long music generation from raw audio with a simple convolutional GAN," *arXiv*, available at https://arxiv.org/abs/2101.04785, 2021.

[12] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *Proc. of International Conference on Learning Representations*, 2019, [dataset available at https://magenta.tensorflow.org/datasets/maestro].

[13] N. Tokui, "Can GAN originate new electronic dance music genres? – generating novel rhythm patterns using GAN with genre ambiguity loss," *arXiv*, available at https://arxiv.org/abs/2011.13062, 2020.

[14] G. Li, S. Ding, and Y. Li, "Novel LSTM-GAN based music generation," in *Proc. of 13th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2021, pp. 1–6.

[15] Z. Chang, G. A. Koulieris, and H. P. H. Shum, "On the design fundamentals of diffusion models: A survey," *arXiv*, available at https://arxiv.org/abs/2306.04542, 2023.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[17] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016.

[18] H. Sun, F. Zhang, and Y. Zhang, "An LSTM and GAN based ECG abnormal signal generator," in *Proc. of Advances in Artificial Intelligence and Applied Cognitive Computing*, Hamid R. Arabnia, Ken Ferens, David de la Fuente, Elena B. Kozerenko, José Angel Olivas Varela, and Fernando G. Tinetti, Eds., Cham, 2021, pp. 743–755, Springer International Publishing.

[19] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," vol. 404.

[20] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*, Cambridge University Press, 2023.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv*, http://arxiv.org/abs/1412.6980, 2014.