IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Generative Music Composition Using GANs

[1]Utsha Sarker, [2]Archy Biswas, [3]Saurabh, [4]Lalit Vaishnav, [5]Myla Vizwal Rathod

[1,2,3,4,5]Apex Institute of Technology (CSE),
[1,2,3,4,5]Chandigarh University, Mohali, India

Abstract : This research paper researches the use of GANs towards creative music making using MIDI sequences. In contrast to conventional techniques, GANs utilize adversarial learning between a generator and discriminator to generate coherent and musically substantial compositions. The envisaged model employs a Dual-directional LSTM-based network for the discriminator and an LSTM-based network for the generator, which allows it to learn intricate temporal dependencies in musical data. The generator generates sequences of notes from the latent space, and the discriminator tests whether they are real or not versus true MIDI sequences. Through the training of the model from a varied set of MIDI files, the system learns to create original compositions that mimic the complex patterns in music. Our method combines data cleaning methods, like MIDI parsing also its sequence stabilizations, to improve the effectiveness of the model. Experimental findings noting also the GAN is able to generate distinctive and contextually relevant musical outputs with success, illustrating the promise of deep learning in advancing automatic music composition. The research identifies some of the chief challenges, including maintaining generator-discriminator dynamics and producing output diversity, while providing guidance for future progress in generative music systems.

IndexTerms **- Generative Adversarial Neural Networks (GANs), Automated Music Generation Systems, Long Short-Term Memory Framework, MIDI Data Sequences, and Sequence Standardization Techniques.**

1 INTRODUCTION

Through a Bi-directional LSTM-based discriminator and an LSTM-based generator, the network can integrate complex temporal dependencies in musical data. The discriminator judges note flows that are synthesized from latent space for their validity against actual ones MIDI sequences. Under different training on MIDI files, the generative model generates new, musically significant pieces, with the promise of deep learning for future advancement of automated music composition and insights into challenges and opportunities improvements.

1.1 Generative Network Based on Adversarial Learning (GAN)

The arrival at scale of artificial intelligence and deep learning has had a negative impact on several activities, including music production, where algorithms attempt to replicate the human imagination. Of the modern techniques, Generative Adversarial Networks (GANs) have been a resounding success at creating realistic data by challenging dual neural networks, namely the synthesizer and the classifier, in a game where the two networks are learned by increasingly dependent on one another's output. GANs initially originated for image production, yet they have been generalized to work on other applications, e.g., music production, where the task is to generate rich, structured information in the form of MIDI files.

1.2 MIDI: Digital Communication Standard for Musical Instruments

MIDI files, which contain musical information in digital format, provide a universal conveyance for music machine learning. They provide neural networks for construction and knowledge of musical parameters like pitch, length, and rhythm that form a key component in creating sound compositions. Under this framework, researchers can train models to detect structures and music patterns and subsequently allow algorithms to learn styles, genres, or even a particular composer's style.

1.3 Short Long-Term Memory (LSTM)

It is also a crucial aspect of music synthesis with the capability to learn sequential dependencies that are crucial in maintaining the stability of musical notes and phrases. LSTMs allow a model to recall previous notes in a sequence, which are particularly appropriate to music synthesis problems where continuity and coherence in the time domain are appropriate.

In spite of the potential demonstrated by LSTMs and GANs in music generation, many difficulties remain. Perhaps the most important challenge is preserving diversity and coherence in output music. Whereas GANs can generate innovative music, they can also fail to preserve musical structure or prevent repetitive designs. In addition, training GANs to produce high-quality sequences that well represent musical harmony is a matter requiring large data sets and heavy computation. One other challenge is the ability of the model to understand both short-range dependencies (such as individual notes) and long-range dependencies (such as musical

phrases or sections) without losing consistency. The paper examines the design, training, and testing of a GAN-based system with LSTM layers incorporated for producing MIDI sequences. We introduce a method to generate new, coherent music pieces and show how GANs, in conjunction with LSTM networks, can produce realistic and musically diverse MIDI files. Our work counteracts structure and coherence issues in music generation, enlarging the field of AI-based music composition.

2 LITERATURE REVIEW

H. Zhang et al. [1] discuss how to apply Generative Adversarial Networks (GANs) to music generation with a focus on their ability to reduce human intervention. It provides the basic principles of GANs, their application in music generation, the problems encountered, and useful insights for future machine learning-based music generation research. Tony el al. [2] suggest utilizing LSTM and GANs for music generation automatically. The LSTM model generates music from input files, and GANs produce piano-like music from midi databases. The generated music's quality is evaluated on the framework of harmony and aesthetics, with potential improvement in continuity of notes. Almeida el al. [3] suggests a music generation framework built on DCGANs. The audio samples are being converted into a usable format into time-frequency forms, the system creates music segments comparable to the dataset. User experiments indicate that generated segments are musically coherent and not mere noise. Li, Ding el al. [4] proposed an LSTM-GAN model-based approach for music creation to overcome issues such as slow computation and long-term dependencies in conventional neural networks. The model integrates LSTM models combined with GANs for better accuracy. They present a novel data cleaning conversion rule for Musical Instrument Digital Interface information and setup the accuracy of the model based on maximum mean discrepancy. Experimental results demonstrate that the model can autonomously produce new and high-quality music.

Maduskar el al. [5] presented a system integrating Generative Adversarial Networks (GANs) with a recurrent autoregressive model for music generation. This approach addresses the challenge of music sequence generation by accurately capturing both global coherence and local musical structures, overcoming the weakness of previous methods such as WaveNet. Dai el al. [6] introduced a deep model fusing Generative Adversarial Networks (GANs) and Long Short-Term Memory (LSTM) networks are utilized to generate music. LSTM serves as the core component in both the synthesizer and the classifier, allowing the system to create music sequences with patterns closely resembling the input data, thereby demonstrating the effectiveness of combining GAN and LSTM for music generation. Arora et al. [7] describe the application of a basic Generative Adversarial Network (GAN) for creating multi-instrumental MIDI compositions. A pre-processing algorithm was developed to simplify MIDI encoding, and the GAN was implemented using PyTorch. While the GAN produced MIDI data, the result was not musically clear and melodic because of the insufficiency of network complexity. Z. Li et al. [8] introduce a new music generation approach, Leak-GAN, which strengthens adversarial learning by enabling the discriminator to better lead the generator. Comparative experiments with an LSTM model indicate that Leak-GAN produces more coherent, natural, and realistic music, as measured by statistical and music theory criteria.

Zheng and Li [9] proposed LA- SAGAN, a novel model combining Generative Adversarial Networks (GANs) and Self-Attention (SA) for real-time, emotion-based piano music generation. The model leverages SA for long-distance dependencies and emotional features, optimizing structure using Learning Automata. Evaluations show significant improvements in diversity, precision, recall, and musical coherence. Adhikary et al. [10] explore the use of Wasserstein Generative Adversarial Networks (WGANs) for generating Indian classical music based on raga. Using Musical Instrument Digital Interface piano music as input, the WGAN was trained using data of classical music of India. It is shown by the results that the created music is acoustically same to the original human-generated music. Huang et al [11] proposes an LSTM-based GAN for multitrack symbolic music composition to improve temporal correlation and musicality. By using LSTM networks in the classifier and adding a controller between the producer and discriminator, this method enhances the synthesizer's power to create authentic music, as demonstrated through experimental validation. Toh and Sourin [12] propose using a Deep Convolutional Generative Adversarial Network (DCGAN) for music generation with dynamics. By encoding MIDI data with piano-roll representation, the DCGAN learns the distribution of music elements like pitch, time, and velocity. The generated music incorporates dynamics and syncopated rhythm, vali- dated through user evaluation. Tang, et al [13] review three popular deep learning models for music generation: Biaxial-LSTM, DeepJ, and MuseGAN. They compare their application scenarios and evaluation methods, addressing the lack of standard algorithms and model evaluation criteria. The study offers a reference for future research in music generation.

2.1 Data Preprocessing

Data preprocessing involves extracting key musical features from MIDI files, encoding these features into numerical series, and formatting them for GAN input. Parsing MIDI Files: MIDI files are parsed to extract fundamental musical components, including individual notes, chords, tempo, and instrument type. MIDI files represent each note and chord as objects with pitch and timing attributes, which are essential for capturing the structure of a musical sequence.

- Feature Extraction: Key features are extracted from each MIDI file:
  - Notes: Each pitch, representing a distinct musical note, is recorded as a string (e.g., "C4").
  - Chords: Chords, or sets of notes played simultaneously, are represented as groups of pitch numbers in string format (e.g. 60, 64, and 67 are assigned to represent a C major chord).
  - Tempo: Tempo, defining beats per minute (BPM), is used to standardize time intervals between notes, providing rhythm.

– Instrument Type: Although not directly used in the generation, the instrument type (e.g., piano, violin) can be retained for a potential multi-instrument generation.

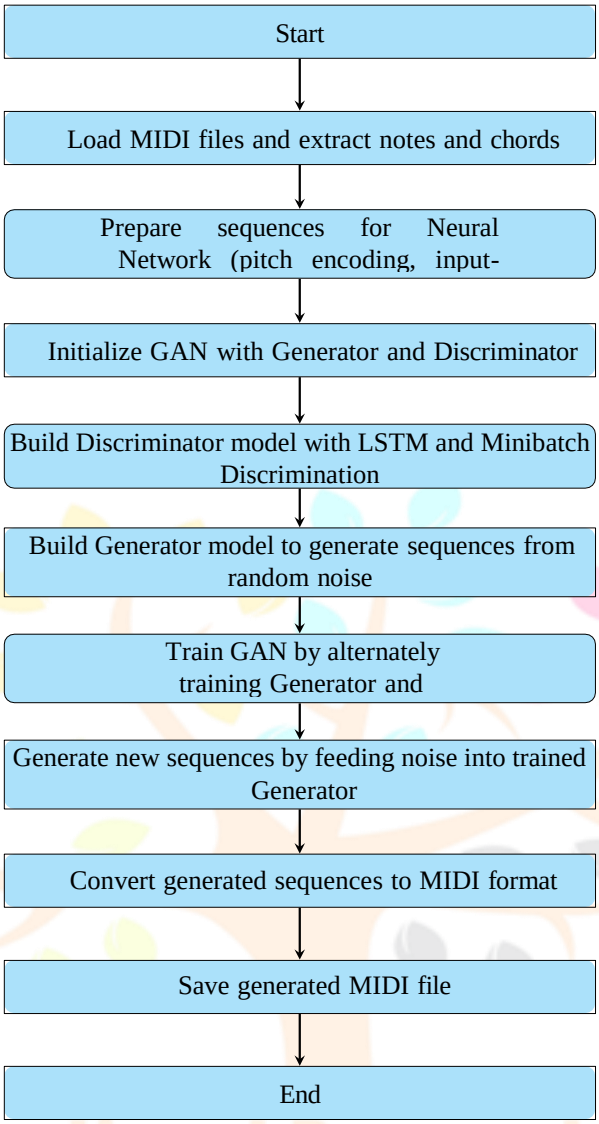

Figure 1. Steps involved in generating new midi file

- Encoding and Vocabulary Creation: Each unique note or chord is mapped to an integer to create a vocabulary. This encoding allows the music sequences to be represented as lists of integers, providing a standardized numerical input for the GAN.
- Sequence Preparation: Using a sliding window approach, each MIDI sequence is divided into overlapping series of a fixed length (e.g., 100 notes), with the last note in each window serving as the target.
- Normalization and Reshaping: The integer sequences are normalized between -1 and 1 to stabilize training and reshaped into a format compatible with LSTM layers in the discriminator.

2.2 GAN Architecture

The GAN architecture is divided into two key networks: the producer and the classifier. The producer synthesizes music sequences from random disturbance, while the classifier assesses while sequences are true or generated. These networks are instructed together by opposing each other, leading to the generation of realistic musical sequences.

2.2.1 Generator Architecture: The synthesizer aims to make music sequences that closely resemble true sequences. It starts with a latent noise vector, which is transformed through multiple dense layers and non-linear activations to generate a structured music sequence.

- Latent Space Input: The synthesizer receives a randomly selected vector drawn from a Gaussian allocation. This vector, denoted as z, has a predefined dimensionality (the "latent dimension") and represents the source of randomness in the generated sequence. $z \sim N(0, 1)$

- Dense Layer Transformations: The noise vector z is passed through fully connected (dense) layers. Each dense layer expands the dimensionality of z, progressively shaping it into a form that resembles the target music sequence.

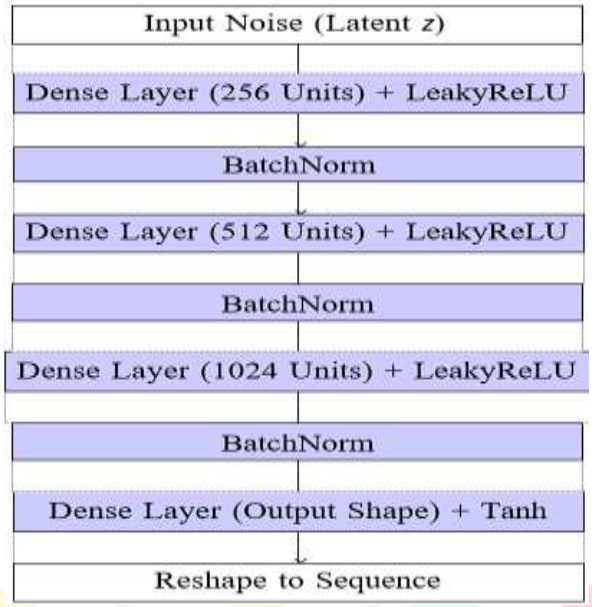

Figure 2.Generator Architecture

Each dense layer is followed by an initiation function, explained as:

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha x, & \text{if } x \leq 0 \end{cases}$$

- Batch Normalization: Batch normalization is applied after each dense layer to normalize the training by stabilizing the outputs. This layer enhances convergence and mitigates problems like internal covariate shift. Given an input x, batch normalization computes:
- Reshaping and Output: The final layer of the generator reshapes the dense layer output into the required sequence format. It uses a tanh activation function to bound values between -1 and 1, preparing the data for the next steps:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{-x} + e^{x}}$$

The generator's objective is to minimize the adversarial loss by "fooling" the discriminator, making it as close to classifying generated sequences as real.

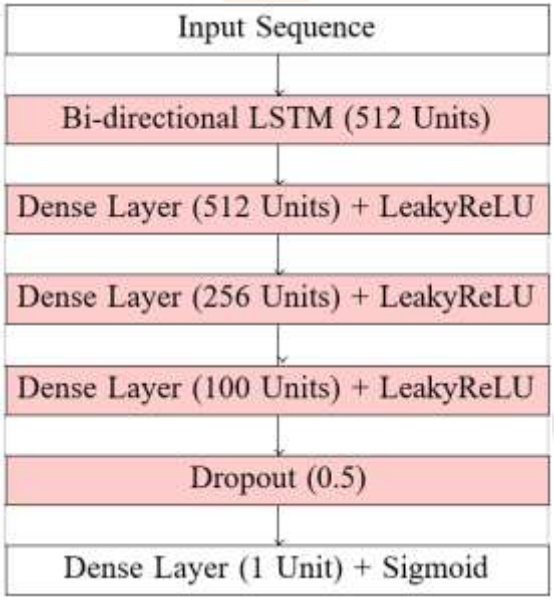

Figure 3. Discriminator Architecture

2.2.2 Discriminator Architecture: The discriminator is structured to classify sequences as real or fake. It uses a series of LSTM and dense layers to capture temporal dependencies within sequences and perform binary classification.

- LSTM Layers: The discriminator starts with LSTM layers that capture the temporal patterns within each sequence. An LSTM unit computes a sequence of outputs as determined by the input and its internal condition. For each timestep t, the LSTM executes the following operations:

– Forgot gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

– Entry terminal:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

– Cell status update:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

– Exit terminal:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

– Hidden state:

$$h_t = o_t \cdot \tanh(C_t)$$

- Bidirectional Layer: To improve context understanding, a bidirectional LSTM layer is used. This layer processes the input series in both backward and forward directions, capturing data from future and past states within the sequence.
- Minibatch Discrimination: Minibatch discrimination is added to prevent mode collapse, encouraging the synthesizer to produce a diverse set of sequences. It introduces small perturbations to the output by calculating a similarity matrix within each batch. For an input matrix M, this operation computes.

$$s_{ij} = \exp(-\gamma |M_i - M_j|^2)$$

Where sij is the similarity between inputs Mi and Mj:

- Binary Classification Layer: The discriminator's final layer uses a sigmoid activation function to classify series as real or generated:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The classifier's goal is to maximize the adversarial loss by correctly distinguishing real sequences from generated ones.

2.2.3 Learning Process: The learning process includes adversarial training, where the synthesizer goals to produce sequences which can "fool" the classifier, while the classifier learns to correctly classify and synthesized sequences. The following steps elaborate on this process, including the loss functions used:

- Adversarial Training Objective: GAN training is an iterative process that updates the producer and classifier networks in alternation. Each iteration of the GAN learning loop aims to optimize the following objectives for both networks:
  – The synthesizer is instructed to reduce the chance that the classifier accurately classifies the synthesized outputs as fake.
  – The classifier is instructed to maximize its ability to tell apart true and synthesized information.

The adversarial loss for the Generative Adversarial Network is formulated as below:

$$L_{\text{GAN}} = E_{\text{real}}[\log(D(\text{real}))] + E_{\text{fake}}\left[\log\left(1 - D(G(\text{noise}))\right)\right]$$

Here the classifier is ()D(), ()G() is the synthesizer, real represents real data, and noise is random input sampled from a distribution.

- Classifier Loss Calculation: The discriminator is instructed on batches of true sequences and synthesized series. The loss for real sequences denoted as L_{real} and for generated sequences, L_{fake}, is explained as follows:

$$L_{\text{real}} = -\frac{1}{N}\sum_{i=1}^{N} \log(D(\text{real}_i))$$

$$L_{\text{fake}} = -\frac{1}{N}\sum_{i=1}^{N} \log\left(1 - D(G(\text{noise}_i))\right)$$

Where batch size is N, and D outputs the chances that a sequence is true.

The total classifier loss LD is:

$$L_D = \frac{1}{2}(L_{\text{real}} + L_{\text{fake}})$$

- Generator Loss Calculation: The generator aims to pro- duce sequences that the classifier classifies as true. Thus, the synthesizer loss, LG, is formulated as:

$$L_D = \frac{1}{2}(L_{\text{real}} + L_{\text{fake}})$$

- Optimization: Generator and discriminator losses are minimized with the use of the Adam optimizer, which optimizes model parameters by backpropagating the gradients from each network's loss function.

2.2.4 Music Sequence Generation and Conversion:

Following training, random noise vectors are input to the trained generator to generate novel sequences. These sequences are then projected back to the musical notes in the original vocabulary and translated into a MIDI form to be played and analyzed.

3 EXPERIMENTAL RESULTS

3.1 Brief Description of Dataset

The data employed herein is composed of MIDI files obtained from different public collections to provide a wide variety of musical genres and styles. Each MIDI file is being read in to extract the chords and notes and subsequently processed to generate sequences appropriate for training. Such sequences encode the detailed patterns and transitions in music that help the model learn complicated musical structures. The overall number of notes and chords defines the size of the vocabulary employed in the model, which has a direct impact on its capacity to produce varied compositions.

For datasets for training and testing, the dataset is categorized into 80% for learning purposes while 20% for validation. This training dataset comprises input sequences of fixed length, the following note as the output target. This configuration enables the model to acquire knowledge temporal relations also in predict future notes, improving its capability to produce well-coherent and contextually correct musical pieces. The validation set is utilized to track performance and avoid overfitting to let the model will generalize well to new datasets.

3.2 Evaluating Performance

3.2.1 Loss Comparison:

The generator loss formula is written as:

$$L^G = -\frac{1}{N}\sum_{i=1}^{N} \log\left(D(G(z_i))\right)$$

Explanation of Terms:

- L_G: synthesizer loss, measuring how well the synthesizer is fooling the classifier.
- N: Total number of samples.
- $D(G(z_i))$: The output of the discriminator when given a generated sample $G(z_i)$.
- $G(z_i)$: The output of the synthesizer given noise entry.
- z_i.
- log: Natural logarithm, used to calculate the adversarial component.

• Discriminator Loss Formula The discriminator loss formula consists of two components:

$$L_D = -\frac{01}{N}\sum_{i=1}^{N}\left[\log(D(x_i)) + \log\left(1 - D(G(z_i))\right)\right]$$

Explanation of Terms:

- L_D: classifier loss, indicating how well the classifier differentiates between true and synthesized data.
- $D(x_i)$: Output of the classifier for a true sample x_i.
- $1 - D(G(z_i))$: The output indicating how much the discriminator believes a generated sample is fake.
- log: Natural logarithm, ensuring that the loss function penalizes incorrect predictions.

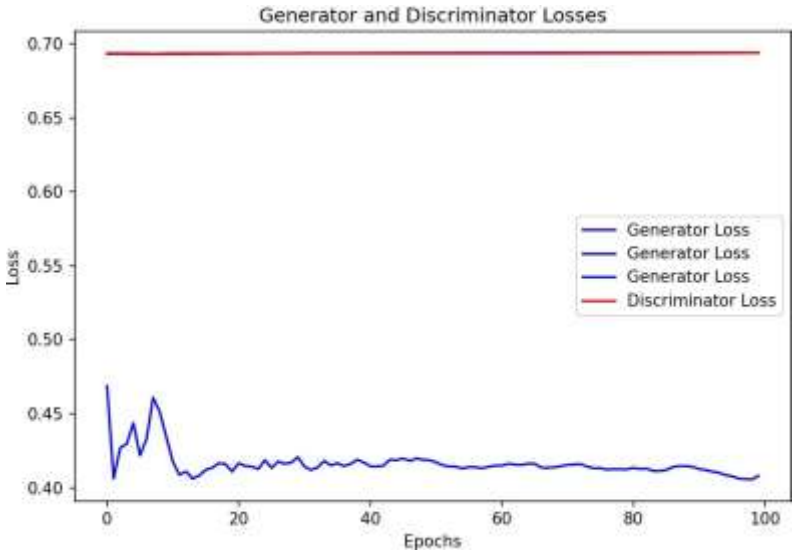

Figure 4. Discriminator loss X generator loss

Figure 4 illustrates the loss metrics for both the synthesizer and the classifier throughout the learning process. The generator loss (depicted in blue) and the discriminator loss (shown in red) are plotted across 100 training epochs. Initially, both models exhibit fluctuations in their respective loss values, indicating the adversarial nature of their learning. The classifier's loss stabilizes at higher level,

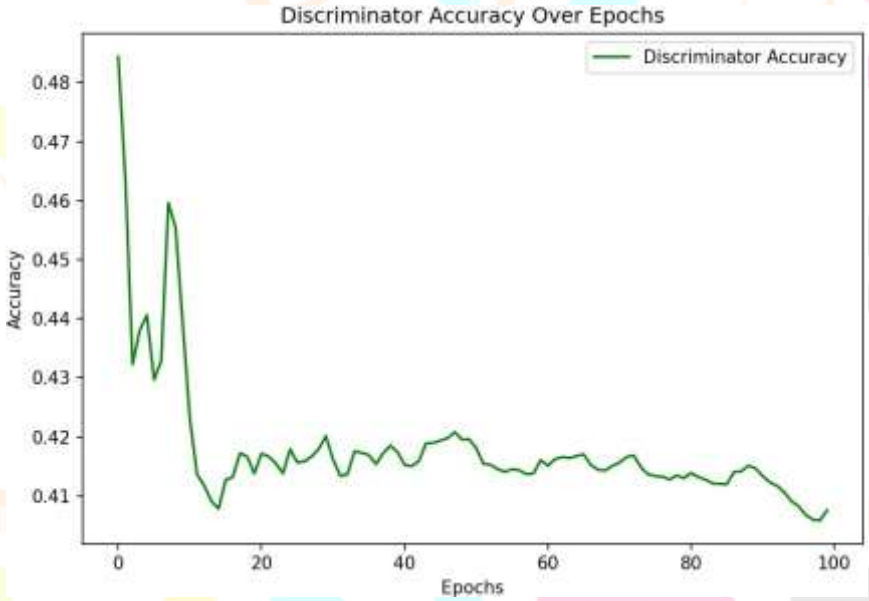

Figure 5. Discriminator accuracy over epochs

While the synthesizer's loss gradually drops and converges as learning progresses. This trend demonstrates the generator's learning curve as it becomes more adept at producing realistic musical sequences, effectively reducing its loss over time. The stable discriminator loss suggests that it consistently distinguishes between real and generated sequences, maintaining a balanced training dynamic.

3.2.2 Discriminator Accuracy over epochs:

- Discriminator accuracy:

$$\text{Accuracy}_D = \frac{TN + TP}{FP + TP + TN + FN}$$

Figure 5 presents the efficiency of the classifier over 100 training epochs. The accuracy metric measures how effec- tively the discriminator classifies real and generated sequences. Initially, accuracy values show significant variability, reflect- ing the early stages of training where the generator is still producing suboptimal outputs. Over time, accuracy stabilizes around a certain threshold, indicating that the discriminator maintains a reliable performance level without overfitting. This stable accuracy, combined with the decreasing generator loss from Figure 4, highlights an improvement in the generator's ability to create plausible outputs, achieving a balance where neither the generator nor the discriminator dominates the training process. This equilibrium is crucial for successful GAN training, ensuring the generation of high-quality, realistic musical compositions.

4 CONCLUSION

The research delved into the application of GANs in auto- mated music composition with the help of MIDI sequences and the employment of an LSTM generator and a bi-directional LSTM discriminator. The results demonstrated that LSTM models, in combination with GANs, learned temporal features from the data and were capable of producing music com- positions which were

reminiscent of human composers. The adversarial formulation of the problem ensured that the outputs had some convergence towards a musical timing structure. Training on unstructured MIDI data helped to define the training data's structure and aided in the enhancement. With the inclusion of different MIDI origins, the model extrapolated multifaceted musical genres enabling it to generate cohesive outputs. The music created through GANs was well-structured with rhythm and melody components, indicating that deep learning has immense possibilities in creativity. Important explanations were related to the problem of training dynamics coordination between the generator and the discriminator. Different variables led to the problem of mode collapse, which is a situation where no variety exists in the resultant outputs. The high computational demand of LSTM-based GANs also accentuated the lack of efficient models.

The findings of this study call for an increased application of GANs in music creation and user-specific compositions. Further development of these systems may improve operational constraints and take predictive music to the next level of technological advancement.

REFERENCES

[1] Zhang, H., Xie, L., & Qi, K. (2021). Implement Music Generation with GAN: A Systematic Review. *Proceedings of the 2021 International Conference on Computer Engineering and Application (ICCEA)*, Kunming, China, pp. 352–355. IEEE. https://doi.org/10.1109/ICCEA53728.2021.00075

[2] Tony, S., Subramanian, S., & Sasikumar. (2022). Investigation on Automatic Music Generation Using GAN and LSTM Networks. *Research Square*. https://doi.org/10.21203/rs.3.rs-2376478/v1

[3] Mitra, R., & Zualkernan, I. (2025). Music generation using deep learning and generative AI: a systematic review. *IEEE Access.*

[4] Li, G., Ding, S., & Li, Y. (2021). Novel LSTM-GAN Based Music Generation. *Proceedings of the 2021 13th International Conference on Wireless Communications and Signal Processing (WCSP)*, Changsha, China, pp. 1–6. IEEE. https://doi.org/10.1109/WCSP52459.2021.9613311

[5] Prasad, P. D. S., Tiwari, R., & Saini, M. L. (2023, March). Digital Image Enhancement using Conventional Neural Network. In *2023 2nd International Conference for Innovation in Technology (INOCON)* (pp. 1-5). IEEE.

[6] Zheng, L., & Li, C. (2024). Real-Time Emotion-Based piano music generation using generative adversarial network (GAN). *IEEE Access*, *12*, 87489-87500.

[7] Saini, M. L., Patnaik, A., Sati, D. C., & Kumar, R. (2024, February). Deepfake Detection System Using Deep Neural Networks. In *2024 2nd International Conference on Computer, Communication and Control (IC4)* (pp. 1-5). IEEE.

[8] Arora, S., Dassler, A., Earls, T., Ferrara, M., Kopparapu, N., & Mathew, S. (2022). An Analysis of Implementing a GAN to Generate MIDI Music. *Proceedings of the 2022 IEEE MIT Undergraduate Research Technology Conference (URTC)*, Cambridge, MA, USA, pp. 1–5. IEEE. https://doi.org/10.1109/URTC56832.2022.10002181

[9] Gupta, D., Saini, M. L., Mygapula, S. P. K., Maji, S., & Prabhas, V. (2024, November). Generating Realistic Images through GAN. In *2024 4th International Conference on Technological Advancements in Computational Sciences (ICTACS)* (pp. 1378-1382). IEEE.

[10] Zhang, M. (2025). Advancing deep learning for expressive music composition and performance modeling. *Scientific Reports*, *15*(1), 28007.

[11] Ali, S. A. K., Saini, M. L., Kumar, E. G., & Teja, B. B. (2024, November). Upscaling Images Resolution Using Generative Adversarial Networks. In *2024 4th International Conference on Technological Advancements in Computational Sciences (ICTACS)* (pp. 1344-1349). IEEE.

[12] Mulakala, B., Saini, M. L., Singh, A., Bhukya, V., & Mukhopadhyay, A. (2024, November). Adaptive multi-fidelity hyperparameter optimization in large language models. In *2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS)* (pp. 1-5). IEEE.

[13] Saini, M. L., Telikicharla, R. S., & Sati, D. C. (2024, March). Handwritten English Script Recognition System Using CNN and LSTM. In *2024 IEEE International Conference on Contemporary Computing and Communications (InC4)* (Vol. 1, pp. 1-6). IEEE.

[14] Toh, R. K. H., & Sourin, A. (2021). Generation of Music With Dynamics Using Deep Convolutional Generative Adversarial Network. *Proceedings of the 2021 International Conference on Cyberworlds (CW)*, Caen, France, pp. 137–140. IEEE. https://doi.org/10.1109/CW52790.2021.00030

[15] Tripathi, B., Saini, M. L., Jain, M., Saxena, A., & Pal, N. (2025, February). Fall Detection and Alerts in Hearing Aids for Deaf using Generative AI. In *2025 2nd International Conference on Computational Intelligence, Communication Technology and Networking (CICTN)* (pp. 543-548). IEEE.

[16] Sasidhar, C., Saini, M. L., Charan, M., Shivanand, A. V., & Shrimal, V. M. (2024, November). Image Caption Generator Using LSTM. In *2024 4th International Conference on Technological Advancements in Computational Sciences (ICTACS)* (pp. 1781-1786). IEEE.

[17] Jain, A., Saini, M. L., Bansal, K., & Padhi, A. (2024, November). MRI and CT Scan Images Quality Enhancement Using Generative Adversarial Network. In *2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS)* (pp. 1-6). IEEE.

[18] Sai Ram Pavan, G., Kucharlapati, A. V., Moneesh, N., Abhishek, S., & Anjali, T. (2023, December). Exploring the Potential of GANs, LSTM, and VAEs in Advancing Music Generation. In *International Conference on Data Science, Machine Learning and Applications* (pp. 538-550). Singapore: Springer Nature Singapore.