International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 05 | May - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

MUSIC GENERATION USING NEURAL NETWORK

Ms. Devadharshini C
Department of Artificial Intelligence and Machine Learning
Sri Shakthi Institute of Engineering and Technology, Coimbatore, India

Mr. Nishanth S
Department of Artificial Intelligence and Machine Learning
Sri Shakthi Institute of Engineering and Technology, Coimbatore, India

Mrs. Hemavathi R
Department of Artificial Intelligence and Machine Learning
Sri Shakthi Institute of Engineering and Technology, Coimbatore, India

***Abstract*— This study investigates the utilization of recurrent neural networks (RNNs) for generating music through MIDI files. By encoding musical data into sequences, RNN models are trained to learn patterns and structures inherent in compositions. Through the analysis of MIDI data and the evaluation of generated sequences, the effectiveness of RNNs in autonomously creating cohesive musical pieces is explored, advancing the frontier of AI-driven musical composition.**

Keywords—Music Generation, Long Short-Term Memory, Recurrent Neural Network, MIDI data .

I. INTRODUCTION

Music, an enduring cultural staple, has historically demanded rigorous training and dedication for creation. However, with the emergence of artificial intelligence (AI), this paradigm is shifting. Recurrent neural networks (RNNs) stand out as a powerful tool for music generation, particularly when trained on MIDI files. By learning from existing compositions, RNNs can autonomously craft melodies and harmonies across genres. This innovation not only streamlines the creative process for musicians and producers but also facilitates exploration of new musical territories. Moreover, RNN-generated music from MIDI files finds applications in various domains like film, gaming, and advertising. This project aims to harness RNNs' potential to develop a versatile music generation system, contributing to the evolving AI-music landscape. With MIDI-trained RNNs, music creation becomes more accessible, efficient, and ripe for creative exploration.

II. LITERATURE REVIEW

The landscape of music generation using neural networks, particularly with MIDI files, is rich with diverse approaches and notable advancements. Hadjeres and Pachet's pioneering work introduced DeepBach, leveraging recurrent neural networks (RNNs) to craft Bach-like chorales while respecting global and local musical constraints. MuseGAN, by Dong et al., stands out for its use of generative adversarial networks (GANs) to produce multi-track symbolic music, demonstrating versatility across melody, harmony, and accompaniment. Roberts et al. contributed MusicVAE, harnessing variational autoencoders (VAEs) to model latent representations of MIDI data, facilitating exploration and manipulation of musical attributes. Gauvin, Pachet, and Roy's "Counterpoint by Convolution" showcases the efficacy of convolutional neural networks (CNNs) in generating harmonically coherent melodies reminiscent of Western classical music. Hadjeres, Zakharova, Tentser, and Pachet further extended VAE capabilities with MIDI-VAE, incorporating dynamics and instrumentation modelling to produce expressive compositions with nuanced styles and instrumentation. These studies collectively underline the breadth of neural network methodologies applied to music generation, demonstrating their potential for creative exploration and innovation within the realm of music composition.

III. EXISTING SYSTEM

- Data Collection and Preprocessing :
  Gathering MIDI files or other musical data sources and preprocessing them to a suitable format for input into the neural network.
- Model Architecture :
  Designing an RNN architecture (e.g., LSTM or GRU) capable of learning patterns in music sequences.
- Training :
  Training the RNN model on the collected and preprocessed music data to learn the patterns and structure of music.
- Music Generation :
  Generating new music sequences using the trained RNN model.
- Evaluation :
  Evaluating the generated music sequences using metrics such as musicality, coherence, and similarity to the training data.

DRAWBACKS:

- Limited Data Diversity :
  Without data augmentation techniques, the training data may lack diversity, leading to limited creativity and variation in the generated music.
- Suboptimal Model Performance :
  The model architecture and training parameters may not be optimized, resulting in suboptimal performance in terms of music quality, coherence, and relevance to the input.
- Lack of Customization :
  Without conditional generation, users have limited control over the characteristics of the generated music, hindering customization and personalization according to their preferences.
- Dependency on Data Quantity :
  Training a neural network model requires a large amount of data, and the existing system may struggle to achieve satisfactory results with limited training data availability.
- Minimal User Interaction :
  The system may lack a user-friendly interface and real-time feedback mechanisms, limiting user interaction and engagement with the music generation process.

IV. PROPOSED SYSTEM

- Data Augmentation :
  Incorporating data augmentation techniques to increase the diversity of the training data. This can involve pitch shifting, time stretching, adding noise, etc., to the existing MIDI data.
- Model Architecture Optimization :
  Experimenting with different RNN architectures, hyperparameters, and regularization techniques to improve the model's performance and robustness.
- Conditional Generation :
  Implementing conditional generation where specific musical attributes (e.g., genre, mood, tempo) can be provided as input to the model to generate music tailored to those attributes.
- Transfer Learning :
  Leveraging pre-trained models or fine-tuning models trained on large music datasets like MIDI-MAESTRO or MusicNet to improve performance, especially with limited data availability.
- User Interface Enhancement :
  Developing a user-friendly interface that allows users to interact with the system more intuitively. This can include features such as real-time preview of generated music, parameter adjustment for conditional generation, and visualization of model internals.
- Feedback Mechanism :
  Implementing a feedback mechanism where users can provide feedback on generated music, which can be used to improve future iterations of the model.
- Integration with Music Theory :
  Incorporating music theory principles into the model architecture or post-processing steps to ensure that generated music adheres to certain musical rules and conventions.
- Deployment and Scalability :
  Optimizing the model for deployment in production environments, ensuring scalability and

efficiency, and providing support for batch processing and parallelization.

ADVANTAGES:

- Enhanced Data Diversity :

  Data augmentation techniques increase the diversity of the training data, allowing the model to learn a wider range of musical patterns and styles, leading to more creative and varied music generation.
- Improved Model Performance:

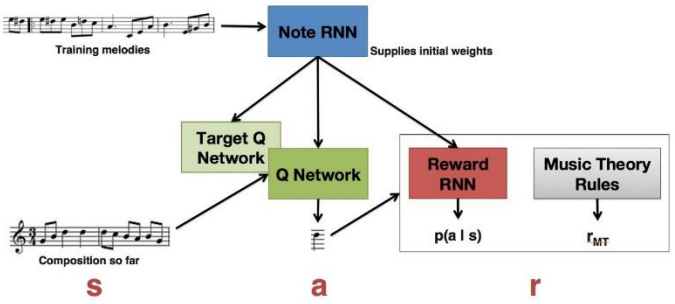
  Experimentation with different architectures, hyperparameters, and regularization techniques can lead to better model performance, resulting in higher-quality, more coherent, and relevant music generation.
- Customization with Conditional Generation:

  Conditional generation enables users to specify desired musical attributes, enhancing customization and personalization according to their preferences, leading to more satisfying user experiences.
- Utilization of Pre-trained Models :

  Leveraging pre-trained models or transfer learning from large music datasets accelerates the training process and can lead to better performance, especially with limited training data availability, improving the overall efficiency of the system.
- Enhanced User Experience :

  A user-friendly interface with real-time preview and parameter adjustment features improves user interaction and satisfaction with the system, fostering greater engagement and enjoyment of the music generation process.

V. MODEL DESIGN



HARDWARE REQUIREMENTS:

- CPU type : Intel Pentium 4
- Clock speed : 3.0 GHz
- Ram size : 4 GB
- Hard disk capacity : 100 GB
- Monitor type : 15 inch colour monitor
- Keyboard type : internet keyboard

SOFTWARE REQUIREMENTS:

- Operating system : Windows OS
- Front End : PYTHON
- Tool : GOOGLE COLAB

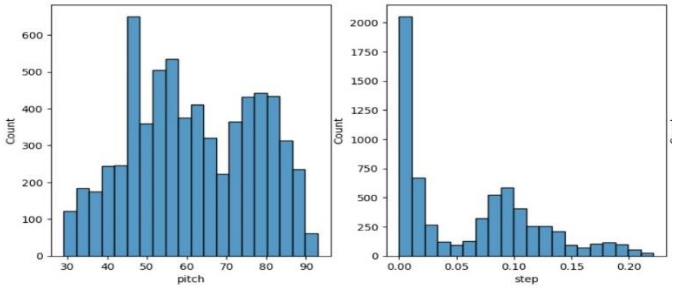
VI. RESULT AND DISCUSION

EXTRACT THE NOTES:

	pitch	start	end	step	duration
0	36	0.992188	1.039062	0.000000	0.046875
1	36	1.165365	1.210938	0.173177	0.045573
2	52	1.183594	1.222656	0.018229	0.039062
3	48	1.187500	1.218750	0.003906	0.031250
4	43	1.191406	1.226562	0.003906	0.035156

It may be easier to interpret the note names rather than the pitches, so you can use the function below to convert from the numeric pitch values to note names. The note name shows the type of note, accidental and octave number (e.g. C#4).

CHECKING THE DISTRIBUTION OF VARIABLE:

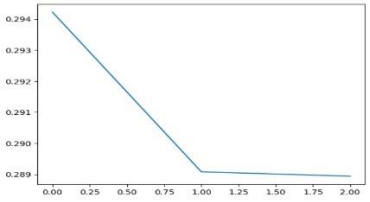


CREATE AND TRAIN THE MODEL:

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 25, 3)]	0	[]
lstm (LSTM)	(None, 128)	67584	['input_1[0][0]']
duration (Dense)	(None, 1)	129	['lstm[0][0]']
pitch (Dense)	(None, 128)	16512	['lstm[0][0]']
step (Dense)	(None, 1)	129	['lstm[0][0]']

Total params: 84354 (329.51 KB)
Trainable params: 84354 (329.51 KB)
Non-trainable params: 0 (0.00 Byte)

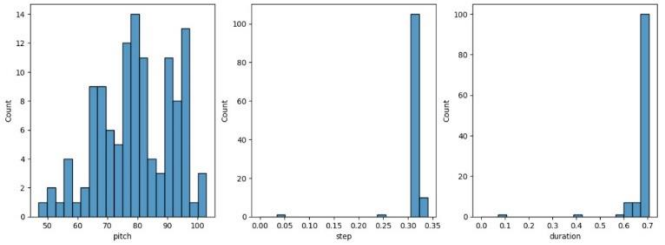
The model will have three outputs, one for each note variable. For step and duration, you will use a custom loss function based on mean squared error that encourages the model to output non-negative values.



GENERATING THE NOTES:

	pitch	step	duration	start	end
0	47	0.044194	0.072837	0.044194	0.117031
1	80	0.251788	0.411856	0.295982	0.707839
2	79	0.339907	0.565379	0.635889	1.201268
3	84	0.342655	0.600813	0.978544	1.579357
4	95	0.340998	0.608585	1.319542	1.928128
5	76	0.340864	0.608600	1.660406	2.269006
6	50	0.336440	0.622982	1.996847	2.619829
7	95	0.325979	0.647549	2.322826	2.970374
8	84	0.335191	0.623230	2.658017	3.281247
9	91	0.331876	0.634524	2.989893	3.624417

VISUALIZING THE GENERATED NOTES:



VII. CONCLUSION

In summary, neural networks trained on MIDI data offer a transformative approach to music generation, enabling the autonomous creation of diverse compositions across genres. From recurrent and convolutional models to generative adversarial networks, these systems streamline the creative process for musicians and producers. Integration of dynamics and instrumentation modelling enhances realism and expressiveness, expanding applications in style transfer and live performance. Challenges such as scalability and diversity persist but do not detract from the profound impact of AI-driven composition on the music industry.

VIII. REFERENCES

- Curtis Hawthorne and al. Enabling factorized piano music modelling and generation with the maestro dataset. In International Conference on Learning Representations (ICLR), 2019.
- Cheng-Zhi Anna Huang and al. An improved relative self-attention mechanism for transformer with application to music generation. In arXiv, 2018.
- Aaron Van den Oord and al. Wavenet: A generative model for raw audio. In SSW, 2016.
- music21: a toolkit for computer aided musicology. https://web.mit.edu/music21/.
- Musicautobot. https://github.com/bearpelican/musicautobot.
- Raffel Colin and al. Feed forward networks with attention can solve some long-term memory problems. In arXiv, 2016.