Automatic Melody Generation Using LSTM Neural Networks

Sheeba P.S.
Dept. of Computer Engineering
Lokmanya Tilak College of Engineering, Navi Mumbai, India.
sheebaps@ltce.in

***Abstract*— Recent advancements in artificial intelligence and deep learning have significantly influenced the field of computational creativity, particularly in automatic music composition. This paper presents a music generation system based on Long Short-Term Memory (LSTM) neural networks, a specialized type of recurrent neural network capable of learning long-term temporal dependencies in sequential data. The proposed model is trained on musical note sequences to capture melodic structures, rhythmic patterns, and harmonic relationships, enabling it to generate original music that closely resembles the style of the training dataset. The study outlines the preprocessing of musical data, the architecture of the LSTM model, the training methodology, and the music generation process. In addition, the impact of model parameters on the creativity, diversity, and coherence of the generated compositions is analyzed. Experimental results demonstrate that the LSTM-based approach is capable of producing musically consistent and aesthetically pleasing compositions, highlighting its potential for applications in automatic music composition, background music generation, educational tools, and computer-assisted creative systems. The study also discusses the opportunities and challenges associated with AI-generated music, including originality, artistic creativity, and ethical considerations.**

***Keywords*— Long Short-Term Memory (LSTM), Music Generation, Deep Learning, Artificial Intelligence, Recurrent Neural Networks (RNN), Automatic Music Composition, Sequence Modeling.**

I. INTRODUCTION

Music is often regarded as a universal language that transcends cultural and linguistic boundaries, providing emotional expression, relaxation, and entertainment. The creation of music involves the harmonious arrangement of sounds with varying frequencies, rhythms, and melodies to produce aesthetically pleasing compositions. Traditionally, composing music requires creativity, musical knowledge, and extensive experience. However, recent advancements in artificial intelligence (AI) and deep learning have enabled computers to learn musical structures and generate original compositions automatically.

Automatic music generation has become an active area of research within computational creativity. By learning patterns from existing musical compositions, deep learning models can produce new melodies that resemble human-created music while maintaining originality. Such systems have applications in music composition, game and film soundtracks, virtual assistants, educational tools, and computer-aided music production.

This work focuses on generating music automatically using a multi-layer Long Short-Term Memory (LSTM) neural network. LSTM is a specialized form of Recurrent Neural Network (RNN) designed to model sequential data by effectively capturing long-term dependencies. Unlike conventional RNNs, which often suffer from the vanishing gradient problem and have limited memory of earlier inputs, LSTM networks employ memory cells and gating mechanisms that enable them to retain relevant information over extended sequences. This capability makes LSTM particularly suitable for music generation, where the relationships among notes, rhythm, and harmony span long intervals.

The proposed system is trained using ABC music notation, a simple text-based notation system in which musical notes are represented by the letters A–G along with symbols that describe rhythm, duration, and other musical attributes. The simplicity of ABC notation makes it well suited for machine learning applications, as musical compositions can be represented as sequential textual data. After training, the model generates new musical sequences in ABC notation, which can be rendered into playable sheet music and audio using tools such as abcjs Quick Editor.

To improve the diversity and quality of the generated compositions, the training dataset consists of ABC sheet music collected from five composers and incorporates melodies written for two different musical instruments. The combined dataset enables the model to learn a wider range of melodic, rhythmic, and harmonic patterns. A multi-layer LSTM architecture with multiple hidden layers and memory cells is employed to capture these complex musical structures more effectively than a single-layer LSTM model.

The objective of this research is to develop an intelligent music generation system capable of producing original, melodious, and coherent musical compositions rather than simply reproducing the training data. By learning underlying musical patterns and stylistic characteristics, the proposed model generates compositions of reasonable quality, longer duration, and improved musical coherence. The study demonstrates the effectiveness of multi-layer LSTM networks for automatic music generation and highlights their potential in supporting composers, content creators, and AI-assisted creative applications.

II. RESEARCH OBJECTIVE

The primary objective of this research is to investigate the effectiveness of Long Short-Term Memory (LSTM) Neural Networks for automatic music generation using ABC music notation. The proposed system is designed to learn musical patterns from a dataset of ABC notations and generate original musical compositions by predicting subsequent notes in a sequence. Starting from an initial musical note or sequence, the model iteratively generates new notes until a complete and coherent musical composition is produced.

The specific objectives of this research are as follows:

A. Data Encoding and Sequence Learning

To preprocess and encode ABC music notation into a numerical format suitable for training the LSTM neural network. The encoded sequences enable the model to learn melodic, rhythmic, and structural patterns present in the training dataset, facilitating the generation of musically meaningful compositions.

B. Model Training and Hyperparameter Optimization

To develop a multi-layer LSTM architecture and optimize its performance through systematic tuning of key hyperparameters, including the number of hidden layers, memory units, learning rate, batch size, and sequence length. The objective is to improve the model's ability to capture long-term musical dependencies and generate coherent musical sequences.

C. Generation of Original Musical Compositions

To design a music generation framework capable of producing unique and creative musical compositions by iteratively predicting the next musical note in a sequence. The generated music should demonstrate continuity, melodic consistency, and structural coherence while avoiding direct reproduction of the training data.

D. Performance Evaluation

To evaluate the quality of the generated music based on musical characteristics such as melody, rhythm, harmony, and grammatical correctness of the ABC notation. The evaluation assesses the model's ability to produce aesthetically pleasing, musically consistent, and structurally valid compositions that adhere to established musical principles.

Through these objectives, the study aims to demonstrate the potential of LSTM-based deep learning models as effective tools for computational creativity and intelligent music composition.

III. LITERATURE SURVEY

Several researchers have explored the application of deep learning techniques for automatic music generation. Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, have demonstrated significant potential in learning sequential musical patterns and generating coherent compositions. A summary of relevant literature is presented below.

A. LSTM-Based Music Generation System – Sanidhya Mangal, Rahul Modak, and Poorva Joshi (2019)

Mangal et al. proposed a music generation system based on Recurrent Neural Networks (RNNs), specifically employing a single-layer Long Short-Term Memory (LSTM) architecture. The model was trained on MIDI files containing pop music and was capable of learning harmonic and melodic relationships between musical notes. By retaining information from previous sequences, the LSTM network generated polyphonic music with reasonable coherence, demonstrating the effectiveness of LSTM networks for sequence-based music generation. However, the use of a single-layer architecture limited the model's ability to capture more complex musical structures and long-term dependencies [1].

B. Music Generation with Deep Learning Algorithms – Eftim Zdravevski, Petre Lameski, and Andrea Kulakov (2018)

Zdravevski et al. developed a multi-layer LSTM Recurrent Neural Network, a feedforward neural network, and an LSTM Encoder–Decoder architecture for music generation. Their objective was to generate polyphonic musical compositions of approximately 60 seconds in duration. Although the proposed models successfully learned musical sequences, they were unable to consistently generate long, coherent compositions. The authors highlighted several limitations related to sequence continuity and model complexity, emphasizing the need for further improvements in deep learning architectures for music generation [2].

C. Deep Learning for Music Generation: Challenges and Directions

This study presents a comprehensive review of deep learning techniques applied to automatic music generation. The authors discuss the strengths of deep learning models in learning musical patterns from existing datasets while identifying several challenges, including limited creativity, lack of user control over melody and harmony, and insufficient interaction between AI systems and human composers. The paper suggests future research directions involving controllable music generation, improved model interpretability, and enhanced collaboration between artificial intelligence and musicians [3].

D. A Lightweight Deep Learning-Based Approach for Jazz Music Generation in MIDI Format

This research introduces a lightweight deep learning framework for generating jazz music in MIDI format. The proposed model is trained on a collection of jazz MIDI files and is capable of producing new musical compositions that preserve the stylistic characteristics of the training data. The study also demonstrates the ability to specify musical instruments during generation, making the system suitable for resource-constrained environments and real-time music generation applications [4].

E. Music Generation Using Deep Learning Techniques

This work presents a comparative analysis of two popular deep learning approaches for music generation: Long Short-Term Memory (LSTM) networks and Generative Adversarial Networks (GANs). Both models were trained using MIDI datasets encoded as note and chord sequences. Experimental results indicate that LSTM networks outperform GANs in capturing sequential musical dependencies and generating coherent musical compositions. The study concludes that LSTM-based models are more effective for learning musical styles and producing melodically consistent music [5].

The reviewed literature demonstrates that LSTM-based architectures remain one of the most effective deep learning approaches for automatic music generation due to their ability to model long-term temporal dependencies in musical sequences. However, challenges such as generating longer coherent compositions, enhancing creativity, improving user control, and increasing musical diversity remain active research areas. These observations motivate the development of the proposed multi-layer LSTM-based music generation system, which aims to generate original and melodically consistent compositions using ABC music notation.

IV. Implementation

A. Recurrent Neural Network (RNN):

Recurrent Neural Networks (RNNs) are a class of artificial neural networks specifically designed to process sequential or time-dependent data. Unlike conventional feedforward neural networks, which process each input independently, RNNs retain information from previous inputs through internal memory. This enables the network to capture temporal dependencies and contextual relationships within a sequence.

Human cognition naturally relies on previous experiences and contextual information rather than processing each event independently. Similarly, when reading a sentence, the meaning of each word is understood based on the words that precede it. Traditional neural networks lack this capability because they do not maintain information from earlier inputs, making them unsuitable for tasks involving sequential data such as speech recognition, natural language processing, time-series prediction, and music generation.

RNNs overcome this limitation by incorporating recurrent connections, which create feedback loops within the network. At each time step, the hidden state of the network is updated using both the current input and the hidden state from the previous time step. This mechanism allows information to persist across the sequence and enables the model to learn dependencies between past and present inputs.

Despite their ability to model sequential information, conventional RNNs suffer from the vanishing gradient and exploding gradient problems during training. As the sequence length increases, the network struggles to retain

information from earlier time steps, limiting its ability to learn long-term dependencies. This drawback significantly affects applications such as music generation, where relationships among musical notes extend over long sequences.

To address these limitations, Long Short-Term Memory (LSTM) networks were introduced. LSTM networks incorporate memory cells and gating mechanisms that selectively retain or discard information, enabling them to capture long-term dependencies more effectively. Consequently, LSTMs have become one of the most widely used architectures for sequence generation tasks, including automatic music composition, speech processing, and language modeling.

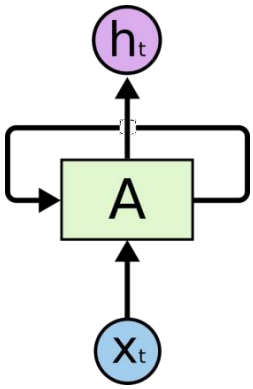

FIG 1: SIMPLE RNN

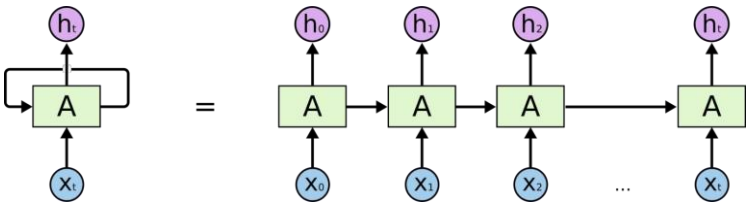

FIG 2: UNROLLED LOOP

B. Architecture of Recurrent Neural Networks

In the above diagram, a chunk of neural network, A looks at some input xt and outputs a value ht. A loop allows information to be passed from one step of the network to the next. These loops make recurrent neural networks seem kind of mysterious. However, if you think a bit more, it turns out that they aren't all that different than a normal neural network. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. Consider what happens if we unroll the loop. This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data. And they certainly are used! In the last few years, there have been incredible success applying RNNs to a variety of problems: speech recognition, language modeling, translation, image captioning… The list goes on. I'll leave discussion of the amazing feats one can achieve with RNNs to Andrej Karpathy's excellent blog post, The Unreasonable Effectiveness of Recurrent Neural Networks. But they really are pretty amazing. Essential to these successes is the use of "LSTMs," a very special kind of recurrent neural network which works, for many tasks, much better than the standard version. Almost all exciting results based on recurrent neural networks are achieved with them. It's these LSTMs that this essay will explore.

C. The Problem of Long-Term Dependencies

One of the major advantages of Recurrent Neural Networks (RNNs) is their ability to utilize information from previous time steps while processing sequential data. This capability allows the network to establish relationships between past and present inputs, making RNNs suitable for applications such as speech recognition, language modeling, time-series prediction, and music generation.

In many sequence learning tasks, only recently observed information is required to predict the current output. For example, in a language model, predicting the final word in the sentence "The clouds are in the..." is straightforward because the immediately preceding words provide sufficient context to predict the word "sky." In such situations, where the dependency between relevant information and the prediction is relatively short, conventional RNNs can effectively learn and utilize contextual information.

However, many real-world problems require the network to remember information over much longer sequences. In music generation, for instance, a melody introduced at the beginning of a composition may reappear much later in the piece. Similarly, in language processing, the meaning of a sentence may depend on words that appeared several sentences earlier. Conventional RNNs often struggle to capture these long-term dependencies because information gradually diminishes as it propagates through many time steps.

The primary reason for this limitation is the vanishing gradient problem, which occurs during the training process using backpropagation through time (BPTT). As gradients are propagated backward through multiple time steps, they become increasingly smaller, making it difficult for the network to update the weights associated with earlier inputs. Consequently, the network gradually "forgets" important long-term information. Conversely, gradients may sometimes

grow excessively large, resulting in the exploding gradient problem, which leads to unstable training.

These limitations significantly reduce the effectiveness of conventional RNNs for tasks involving long sequential dependencies. To overcome this challenge, Long Short-Term Memory (LSTM) networks were introduced. LSTMs incorporate specialized memory cells and gating mechanisms that regulate the flow of information, enabling the network to retain relevant information over extended periods while discarding unnecessary data. As a result, LSTM networks are considerably more effective than traditional RNNs for applications requiring long-term memory, including automatic music generation, natural language processing, and speech recognition.

Thus, the ability of LSTM networks to learn long-term dependencies makes them the preferred architecture for generating coherent and musically consistent note sequences.

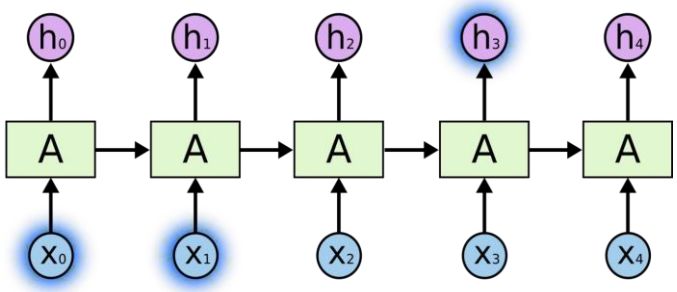

FIG 3: RNNS LEARN USING PAST INFORMATION

While conventional Recurrent Neural Networks (RNNs) perform well when relevant information is located close to the prediction point, they struggle when important contextual information is separated by long intervals. In many real-world sequence modeling tasks, the network must remember information that appeared much earlier in the sequence to make an

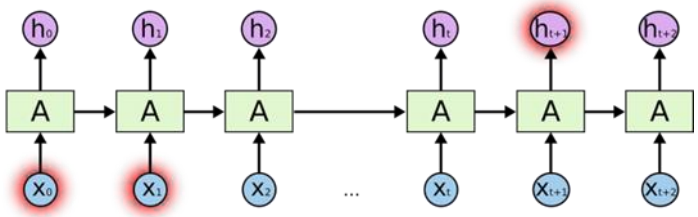

FIG 4: RNNS, UNABLE TO LEARN INFORMATION DUE TO LARGE GAPS

accurate prediction. For example, consider the sentence: "I grew up in France... I speak fluent _____." lthough the words immediately preceding the blank suggest that the missing word is the name of a language, determining the correct language requires remembering the earlier word "France." Since the relevant information is located much earlier in the sequence, the network must preserve this context over many time steps. Similar long-range dependencies frequently occur in music generation, where themes, motifs, or chord progressions introduced at the beginning of a composition may reappear much later.

As the distance between the relevant information and the prediction point increases, conventional RNNs become increasingly ineffective at preserving the necessary context. During training, the repeated multiplication of gradients causes them to either diminish rapidly (vanishing gradients) or grow uncontrollably (exploding gradients). Consequently, the network gradually loses information from earlier time steps and becomes unable to learn long-term relationships within the sequence.

This limitation significantly affects applications involving long sequential data, including language translation, speech recognition, and automatic music composition. In music generation, the inability to remember earlier melodic patterns may result in compositions that lack thematic consistency and harmonic continuity.

To overcome these shortcomings, Long Short-Term Memory (LSTM) networks were developed. LSTMs incorporate memory cells along with input, forget, and output gates that regulate the flow of information through the network. These gating mechanisms enable LSTMs to selectively retain important information for extended periods while discarding irrelevant data. As a result, LSTM networks effectively capture long-term dependencies, making them highly suitable for sequence generation tasks such as music composition, where maintaining musical structure and coherence over long note sequences is essential.

In theory, Recurrent Neural Networks (RNNs) are capable of learning long-term dependencies because information can be propagated through their recurrent connections over multiple time steps. Under ideal conditions and with carefully selected parameters, an RNN can successfully retain information from earlier inputs and use it to make future predictions. However, in practical applications, conventional RNNs rarely achieve this capability due to difficulties encountered during the training process.

The inability of RNNs to learn long-range dependencies was extensively investigated by Hochreiter (1991) and Bengio et al. (1994). Their studies demonstrated that the repeated propagation of gradients through long sequences often leads to the vanishing gradient and exploding gradient problems. As a result, conventional RNNs gradually lose information from earlier time steps and fail to capture relationships that span long intervals within a sequence.

D. Long Short-Term Memory (LSTM) Networks

Long Short-Term Memory (LSTM) networks are a specialized type of Recurrent Neural Network (RNN) developed to overcome the limitations of conventional RNNs in learning long-term dependencies. Introduced by Hochreiter and Schmidhuber in 1997, LSTMs were specifically designed to retain important information over long sequences while preventing the vanishing gradient problem. Since their introduction, LSTM networks have been further refined and have become one of the most widely used deep learning architectures for sequential data processing.

Unlike traditional RNNs, which often struggle to preserve information over extended time intervals, LSTMs are explicitly designed to remember relevant information for long periods. This capability enables them to model complex sequential relationships effectively, making them highly suitable for applications such as music generation, speech recognition, natural language processing, machine translation, handwriting recognition, and time-series forecasting.

Similar to conventional RNNs, LSTMs consist of a chain of repeating neural network modules through which information flows sequentially. In a standard RNN, each repeating module typically consists of a single hidden layer with a nonlinear activation function, such as the tanh activation. While this simple architecture is effective for modeling short-term dependencies, it cannot efficiently retain information across long sequences.

In contrast, each repeating module in an LSTM contains a memory cell and a set of gating mechanisms that regulate the flow of information. These gates determine which information should be stored, updated, or discarded, enabling the network to maintain long-term contextual information while eliminating irrelevant data. The three primary gates of an LSTM cell are:

Forget Gate: Determines which information from the previous cell state should be discarded.
Input Gate: Selects new information that should be stored in the memory cell.
Output Gate: Controls the information passed from the memory cell to the next hidden state.

The combination of these gates allows LSTM networks to preserve important information over many time steps, thereby overcoming the limitations of conventional RNNs. This architecture enables LSTMs to capture complex temporal patterns and long-range dependencies that are essential for sequence generation tasks.

In the context of automatic music generation, LSTM networks learn the relationships among notes, rhythms, chords, and musical phrases from training data. Once trained, the model can generate original musical compositions by predicting one note at a time while maintaining melodic coherence, rhythmic consistency, and harmonic structure. Consequently, LSTMs have become the preferred deep learning model for music generation due to their ability to produce musically meaningful and coherent sequences over extended durations.

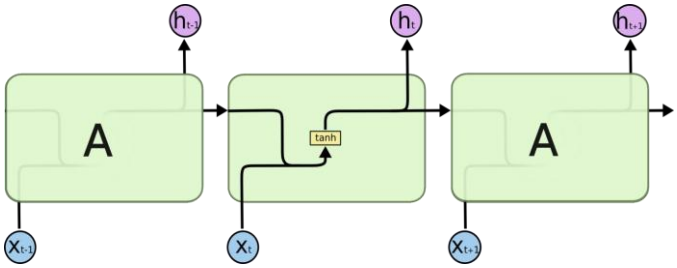

FIG 5: RECURRING NEURAL NETWORK

The repeating module in a standard RNN contains a single layer.

LSTMs also have this chainlike structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

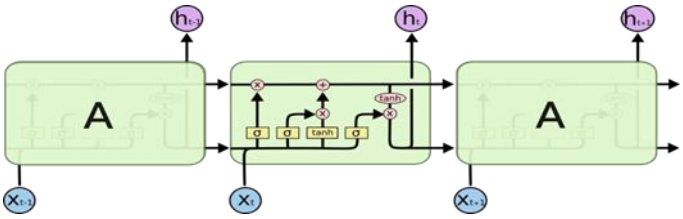

FIG 6: THE REPEATING MODULE IN AN LSTM CONTAINS FOUR INTERACTING LAYERS

Before understanding the detailed operation of an LSTM cell, it is useful to become familiar with the notation commonly used in its architectural diagram. An LSTM network consists of several interconnected components through which information flows sequentially. The diagram illustrates how data is processed, stored, and transferred between successive time steps.

In an LSTM architecture, each line represents the flow of an entire vector rather than a single scalar value. These vectors carry information from the output of one computational unit to the inputs of subsequent units, enabling the propagation of both the hidden state and the cell state throughout the network.

The various symbols used in the LSTM diagram have the following meanings:

Yellow rectangular blocks represent neural network layers that contain trainable parameters. These layers perform mathematical transformations on the input data using learned weights and activation functions.

Pink circular nodes represent element-wise (pointwise) operations, such as vector addition, element-wise multiplication, or other basic mathematical operations performed independently on each vector element.

Merging lines indicate the concatenation of multiple vectors into a single combined vector before further processing by the network.

Forking lines indicate that the same information is copied and transmitted simultaneously to multiple components within the LSTM cell, allowing different gates to operate on identical input data.

This notation provides a clear visualization of how information flows through the LSTM architecture. The interaction between neural network layers, memory cells, and gating mechanisms enables the network to selectively retain, update, and discard information during sequence processing.

Understanding these symbols is essential for interpreting the internal operation of an LSTM cell. The subsequent sections explain each component of the LSTM architecture—including the forget gate, input gate, cell state, and output gate—in detail, illustrating how they collectively enable the network to learn long-term dependencies and generate coherent sequential data such as musical note sequences.

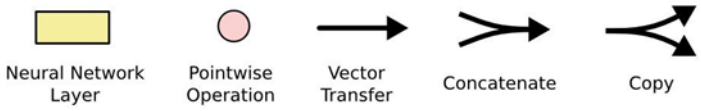

FIG 7: LEGEND

E. Music21

Music21 is an open-source Python toolkit developed for computer-aided musicology, music analysis, and symbolic music processing. It provides a comprehensive framework for reading, analyzing, manipulating, and generating musical data in various formats, including MIDI, MusicXML, ABC notation, and other symbolic music representations. Due to its extensive functionality, Music21 has become one of the most widely used libraries for music-related machine learning and deep learning applications.

In the proposed music generation system, Music21 plays a crucial role in both the preprocessing and post-processing stages. During preprocessing, the toolkit is used to parse the input music dataset and extract note and chord sequences required for training the Long Short-Term Memory (LSTM) neural network. The extracted musical elements are then encoded into a numerical format suitable for sequence learning.

After the LSTM model generates new musical note sequences, Music21 converts the predicted output back into symbolic musical notation. It creates corresponding Note and Chord objects and assembles them into a complete musical score. Finally, the generated composition can be exported as a MIDI or MusicXML file, allowing it to be played, visualized, or further edited using standard music software.

F. Keras

Keras is a high-level, open-source Deep Learning Application Programming Interface (API) designed to simplify the development, training, and deployment of neural network models. It provides an intuitive interface for building deep learning architectures and is fully integrated with TensorFlow, enabling efficient implementation of complex machine learning algorithms with minimal code. In this research, Keras is used to design and implement the multi-layer LSTM model for automatic music generation. The encoded musical note sequences are supplied as input to the network, which learns temporal dependencies and musical patterns during the training phase. After training, the model predicts successive musical notes to generate new compositions. Owing to its simplicity, efficient TensorFlow integration, and extensive support for deep learning algorithms, Keras serves as an ideal framework for developing the proposed music generation system.

G. Model Training

This section describes the dataset used for training the proposed LSTM model, the preprocessing techniques applied to the musical data, and the overall architecture of the neural network. The objective of the preprocessing stage is to convert symbolic music into a numerical representation that can be effectively learned by the LSTM network.

H. Data Set

The proposed music generation system is trained using a collection of piano music compositions, primarily comprising Final Fantasy soundtrack pieces. These compositions were selected because they contain rich melodic structures, expressive harmonies, and diverse musical patterns, making them well suited for training a deep learning model capable of generating aesthetically pleasing music. In addition, the large number of available compositions provides sufficient training data for the LSTM network to learn meaningful musical relationships.

Although the experiments presented in this work use Final Fantasy piano music, the proposed methodology is not restricted to this dataset. Any collection of single-instrument MIDI files, such as piano, violin, flute, or guitar music, can be used to train the model. Using music from a single instrument simplifies the learning process by reducing the complexity associated with multiple simultaneous instrument tracks.

The musical compositions are stored in MIDI (Musical Instrument Digital Interface) format, which represents music symbolically rather than as recorded audio. MIDI files contain detailed information about musical events, including note pitches, note durations, timing, tempo, velocity, and other performance attributes. These files are parsed using the Music21 Python library to extract sequences of notes and chords that serve as the training data for the LSTM network.

The extracted musical information is then preprocessed by converting notes and chords into numerical representations, generating fixed-length input sequences, and normalizing the data for efficient neural network training. This preprocessing

enables the model to learn melodic, rhythmic, and harmonic relationships within the dataset and subsequently generate original musical compositions.

Figure 8 shows an excerpt of a MIDI file after it has been parsed using the Music21 library. The extracted note and chord sequences illustrate the symbolic representation of the musical data that is provided as input to the LSTM model during training.

```
...
<music21.note.Note F>
<music21.chord.Chord A2 E3>
<music21.chord.Chord A2 E3>
<music21.note.Note E>
<music21.chord.Chord B-2 F3>
<music21.note.Note F>
<music21.note.Note G>
<music21.note.Note D>
<music21.chord.Chord B-2 F3>
<music21.note.Note F>
<music21.chord.Chord B-2 F3>
<music21.note.Note E>
<music21.chord.Chord B-2 F3>
<music21.note.Note D>
<music21.chord.Chord B-2 F3>
<music21.note.Note E>
<music21.chord.Chord A2 E3>
...
```

FIG 8: READING NOTES FROM MUSIC

After parsing the MIDI files using the Music21 library, the musical data is categorized into two primary object types: Notes and Chords. These symbolic representations form the basis of the training dataset used by the LSTM network.

A Note object represents an individual musical note and contains several important attributes:

Pitch: Represents the frequency of the sound and determines how high or low a note is. In Western music notation, pitches are denoted by the letters A, B, C, D, E, F, and G, with optional accidentals (sharps and flats).

Octave: Specifies the register in which the note is played, indicating its position on the musical scale. Notes with the same pitch but different octaves have different frequencies.

Offset: Indicates the temporal position of the note within the musical composition. The offset determines when a note begins relative to the start of the piece and is essential for preserving rhythm and timing.

A Chord object represents a collection of two or more notes that are played simultaneously. Rather than representing individual notes separately, Music21 groups notes occurring at the same time into a single chord object, enabling the model to learn harmonic relationships in addition to melodic patterns.

Since the objective of the proposed system is to generate complete musical compositions, the LSTM model must learn to predict the next note or chord in a sequence based on previously observed musical events. Consequently, the prediction vocabulary includes every unique note and chord encountered in the training dataset. In the dataset used for this study, approximately 352 unique notes and chord combinations were identified. Although this represents a relatively large output space, LSTM networks are well suited for handling such sequential prediction problems due to their ability to model complex temporal dependencies.

In addition to learning the sequence of notes and chords, the model must also capture the temporal structure of the music. Musical compositions are characterized by varying intervals between successive notes. Some passages contain rapid sequences of notes, whereas others include longer pauses or sustained notes. Therefore, accurately representing the offset of each musical event is essential for preserving rhythm and producing natural-sounding compositions.

To capture this temporal information, the offset associated with each note and chord is extracted from the MIDI files using Music21. The offset values indicate the time interval between consecutive musical events, allowing the LSTM network to learn not only melodic and harmonic relationships but also rhythmic patterns. This enables the generated music to exhibit appropriate timing, continuity, and musical coherence.

Figure 9 presents an excerpt from a MIDI file processed using the Music21 library, where each note or chord is displayed together with its corresponding offset value. The offset information illustrates the temporal spacing between successive musical events and serves as an important feature during model training.

```
...
<music21.note.Note B> 72.0
<music21.chord.Chord E3 A3> 72.0
<music21.note.Note A> 72.5
<music21.chord.Chord E3 A3> 72.5
<music21.note.Note E> 73.0
<music21.chord.Chord E3 A3> 73.0
<music21.chord.Chord E3 A3> 73.5
<music21.note.Note E-> 74.0
<music21.chord.Chord F3 A3> 74.0
<music21.chord.Chord F3 A3> 74.5
<music21.chord.Chord F3 A3> 75.0
<music21.chord.Chord F3 A3> 75.5
<music21.chord.Chord E3 A3> 76.0
<music21.chord.Chord E3 A3> 76.5
<music21.chord.Chord E3 A3> 77.0
<music21.chord.Chord E3 A3> 77.5
<music21.chord.Chord F3 A3> 78.0
<music21.chord.Chord F3 A3> 78.5
<music21.chord.Chord F3 A3> 79.0
...
```

FIG 9: NOTE AND CHORD INTERVALS

Analysis of the MIDI dataset revealed that the most frequently occurring offset between consecutive musical events is 0.5 beats. Although the dataset contains notes and chords with varying temporal intervals, the majority of musical events follow this common spacing. This observation allows the temporal representation of the data to be simplified without significantly affecting the overall musical quality.

To reduce the complexity of the learning process, the offset values are excluded from the prediction vocabulary during model training. Instead, the LSTM network focuses solely on learning the sequential relationships among notes and chords. Consequently, the output space consists only of the unique note and chord combinations extracted from the dataset, resulting in a vocabulary of 352 distinct musical events.

Ignoring the offset information simplifies both the data representation and the neural network architecture by reducing the number of possible output classes. Although this approximation limits the model's ability to reproduce subtle rhythmic variations, it has only a minor impact on the generated melodies because the predominant note interval in the dataset remains consistent. The generated compositions therefore preserve the essential melodic and harmonic characteristics while significantly reducing computational complexity.

This simplification enables the LSTM model to concentrate on learning musical patterns and note transitions more effectively, leading to improved training efficiency and stable music generation without substantially compromising the quality of the generated compositions.

Data Preparation

After identifying notes and chords as the primary musical features, the next step is to preprocess the dataset so that it can be used for training the Long Short-Term Memory (LSTM) network. Since neural networks operate on numerical data, the symbolic musical information extracted from MIDI files must be converted into an appropriate machine-readable format.

Initially, each MIDI file is loaded into a Music21 Stream object using the converter.parse() function. The stream object provides access to all musical elements contained within the composition, including notes, chords, tempo, and timing information. The parsed stream is then traversed to extract the sequence of musical events.

For every Note object encountered, the pitch is represented using its standard string notation (e.g., C4, D#5, A3). This notation preserves the essential musical characteristics of each note, including its pitch class and octave, while providing a compact symbolic representation that can later be reconstructed into playable music.

For every Chord object, the individual notes comprising the chord are identified and encoded into a single string. The numerical identifiers (or pitch values) of the constituent notes are concatenated using a delimiter, typically a period (.), resulting in representations such as "60.64.67", which corresponds to a C major chord. This encoding uniquely represents each chord while allowing it to be decoded easily during music generation.

After extracting all note and chord sequences from the dataset, a vocabulary of unique musical events is created. Each unique note or chord is assigned a unique integer identifier, converting the symbolic musical data into numerical form suitable for neural network training. The resulting integer sequences are then used to construct fixed-length input-output pairs, where each input sequence consists of a predefined number of consecutive musical events and the corresponding output represents the next event in the sequence.

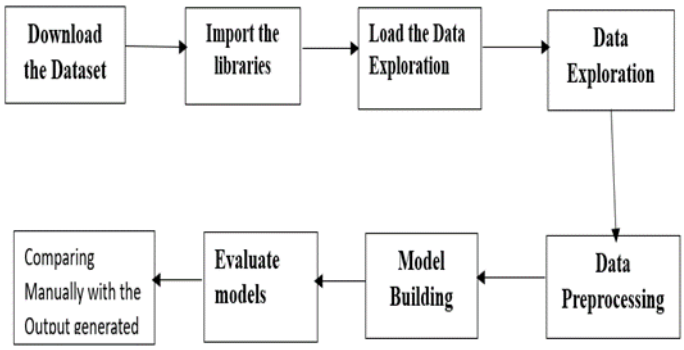

FIG 10: FLOW CHART

V. Simulation Results

The performance of the proposed LSTM-based music generation model was evaluated after training on the prepared MIDI dataset. The model was assessed based on its ability to learn musical patterns and generate coherent note sequences that preserve melodic and harmonic characteristics. The simulation was carried out using the Keras deep learning framework with TensorFlow as the backend, while Music21 was used for MIDI processing and music generation.

The following figures present the results obtained during model training and music generation. These include the training loss curve, generated musical sequences, and the corresponding sheet music or piano-roll representations. The results demonstrate that the LSTM model successfully captures the temporal dependencies among musical notes and chords, enabling it to generate melodically consistent and aesthetically pleasing musical compositions.

The generated output exhibits continuity in rhythm and melody, indicating that the network has effectively learned the underlying musical structures present in the training dataset. Furthermore, the produced compositions are original and are not direct reproductions of the training data, highlighting the model's capability to generate novel musical sequences while maintaining stylistic consistency.

The simulation results confirm the effectiveness of the proposed multi-layer LSTM architecture for automatic music generation and demonstrate its potential for applications in AI-assisted music composition, background music generation, and creative content development.

The performance of the developed model is illustrated in the following figures.

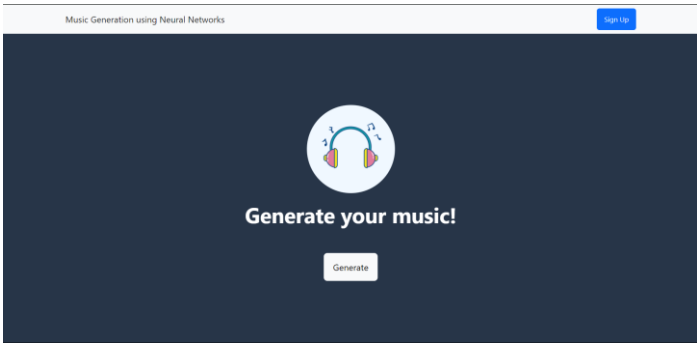

FIG 11: HOME PAGE

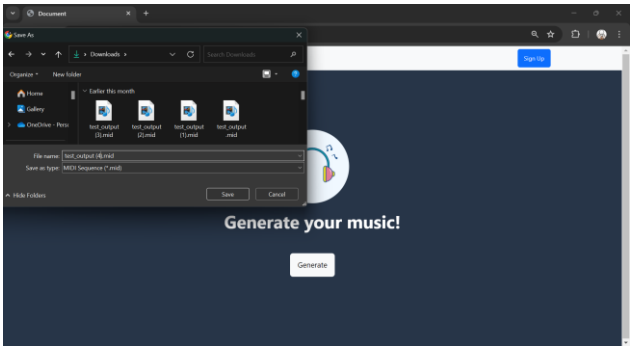

FIG 12: DOWNLOADED MUSIC

VI. Conclusion

This research presented an LSTM-based deep learning framework for automatic music generation using symbolic musical data. The study demonstrated that Long Short-Term Memory (LSTM) neural networks are highly effective in learning temporal dependencies and sequential patterns present in musical compositions. By training the model on MIDI data represented in symbolic notation, the network was able to generate original musical sequences that exhibit melodic coherence, rhythmic consistency, and stylistic similarity to the training dataset.

The proposed framework involved data preprocessing, feature extraction using the Music21 library, sequence encoding, and the development of a multi-layer LSTM model implemented using the Keras deep learning framework. The trained model successfully learned relationships among notes and chords and generated new musical compositions by predicting one musical event at a time. Techniques such as seed-note initialization and sampling strategies were employed to improve the diversity and creativity of the generated music.

The experimental results demonstrate that the proposed model can produce musically meaningful and aesthetically pleasing compositions without directly reproducing the training data. The ability of LSTM networks to retain long-term contextual information enables the generated music to maintain structural continuity and harmonic consistency over extended note sequences.

The proposed system has potential applications in automatic music composition, background music generation, interactive entertainment, educational tools, game and film soundtrack creation, and computer-assisted music composition. It can also serve as a creative assistant for musicians by providing new melodic ideas and compositional inspiration.

Despite its effectiveness, the proposed approach has certain limitations. The quality and diversity of the generated music largely depend on the size and variety of the training dataset. Furthermore, while LSTM networks successfully model sequential dependencies, they may still have difficulty capturing complex hierarchical musical structures present in long compositions. Recent deep learning architectures, such as Transformer-based models and attention mechanisms, offer promising alternatives for addressing these limitations.

Future work may focus on training the model using larger and more diverse musical datasets, incorporating multiple instruments and musical genres, integrating attention mechanisms or Transformer architectures, and developing interactive systems that allow users to control musical style, tempo, mood, and harmony during the generation process.

In conclusion, this research demonstrates that LSTM neural networks provide an effective and practical solution for AI-driven music generation. The study highlights the potential of deep learning to support computational creativity and contributes to the growing field of intelligent music composition. As artificial intelligence continues to advance, AI-assisted music generation is expected to play an increasingly important role in the future of digital music creation, offering new opportunities for composers, performers, and the creative industries.

REFERENCES

[1] Mangal, Sanidhya & Modak, Rahul & Joshi, Poorva. (2019). LSTM Based Music Generation System. IARJSET. 6. 47-54. 10.17148/IARJSET.2019.6508.

[2] Docevski, Marko & Zdravevski, Eftim & Lameski, Petre & Kulakov, Andrea. (2018). Towards Music Generation With Deep Learning Algorithms.

[3] Briot, JP., Pachet, F. Deep learning for music generation: challenges and directions. Neural Comput & Applic 32, 981–993 (2020). https://doi.org/10.1007/s00521-018-3813-6

[4] Yadav, Prasant Singh, et al. "A lightweight deep learning-based approach for Jazz music generation in MIDI format." Computational Intelligence and Neuroscience 2022 (2022).

[5] Prashant Krishnan, V., et al. "Music generation using deep learning techniques." Journal of Computational and Theoretical Nanoscience 17.9-10 (2020): 3983-3987.

[6] E. Zdravevski, P. Lameski, and A. Kulakov, “Music Generation with Deep Learning Algorithms,” *Proceedings of the International Conference on Information Technology and Development of Education (ITRO)*, 2018.

[7] J. Briot, G. Hadjeres, and F. Pachet, “Deep Learning Techniques for Music Generation – A Survey,” Springer International Publishing, 2020.

[8] M. Ali, M. H. Alsharif, S. K. Singh, and M. A. Jan, “A Lightweight Deep Learning-Based Approach for Jazz Music Generation in MIDI Format,” IEEE Access, vol. 9, pp. 99831–99844, 2021.

[9] S. Srivastava and A. Sharma, “Music Generation Using Deep Learning Techniques,” International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), vol. 7, no. 3, pp. 45–51, 2021.

[10] M. Cuthbert and C. Ariza, “Music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data,” Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), pp. 637–642, 2010.