MUSIC GENERATION USING RNN - LSTM

Bonafide record of work done by

SANTHOSHI R (21Z251)

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

19Z006 DEEP LEARNING

of Anna University

OCTOBER 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE 641-004

TABLE OF CONTENTS **PAGE**

1. INTRODUCTION 3
   1.1 OVERVIEW OF THE PROJECT 3
   1.2 PURPOSE AND MOTIVATION OF THE PROJECT 3
   1.3 PROJECT OBJECTIVES 4
2. BACKGROUND AND RELATED WORK 5
   2.1 OVERVIEW OF MUSIC GENERATION 5
   2.2 OVERVIEW RECURRENT NEURAL NETWORKS 5
   2.3 LITERATURE SURVEY 6
3. TECHNOLOGIES AND TOOLS USED 7
4. DATASET 8
5. MODEL ARCHITECTURE 9
   5.1 SYSTEM ARCHITECTURE 9
   5.2 WORKFLOW DIAGRAM 10
6. SYSTEM IMPLEMENTATION 12
   6.1 DEVELOPMENT ENVIRONMENT SETUP 12
   6.2 DATA PREPROCESSING 12
   6.3 MODEL DESIGN AND TRAINING
   6.4 INTERACTIVE USER INTERFACE 13
   6.5 MUSIC GENERATION AND AUDIO CONVERSION 13
   6.6 VISUALIZATION 13
7. RESULTS AND EVALUATION 14
   7.1 OBSERVATION 14
   7.2 SYSTEM EVALUATION 15
8. CONCLUSION 17
9. REFERENCES 18
10. APPENDIX 19

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

In recent years, advancements in artificial intelligence (AI) have expanded beyond traditional applications and ventured into creative domains such as art, literature, and music. This project focuses on the development of a Music Generation System using Recurrent Neural Networks (RNNs) to autonomously compose music. The goal of this project is to build a system that can generate sequences of musical notes based on prior data, producing coherent melodies that mimic human-composed music.

The system is designed to take user input, such as the desired instrument, number of notes, and temperature (a parameter to control the randomness in music generation), providing flexibility in the output. By leveraging deep learning models, specifically RNNs, the project demonstrates how patterns in musical data can be captured and used to predict subsequent notes, creating novel compositions.

The project employs a dataset of MIDI files, which contain musical information such as pitch, note duration, and step size between notes. This data is processed and fed into the RNN model, which learns from the sequences and generates new music by predicting the next note in the sequence. The interactive component of the system, implemented through IPython widgets, allows users to engage with the model in real time, adjusting parameters to influence the generated output.

Overall, this project showcases the potential of machine learning in creative fields and provides an interactive platform for users to explore AI-driven music composition, offering insights into the intersection of technology and creativity.

1.2 PURPOSE AND MOTIVATION OF THE PROJECT

The primary purpose of this project is to explore the potential of Recurrent Neural Networks (RNNs) in the field of music generation, leveraging their ability to model sequential data. Music, by its very nature, is highly sequential, with each note, rhythm, and melody following intricate patterns. RNNs are uniquely suited to capture these temporal dependencies, making them a powerful tool for generating coherent and meaningful musical compositions.

One of the core motivations behind this project is to contribute to the growing intersection of artificial intelligence and creative arts. While AI has made significant strides in areas such as natural language processing and image recognition, its role in music composition represents an exciting frontier. Music is not only mathematically structured but also emotionally resonant, and generating music that is aesthetically pleasing and musically correct presents both a technical and artistic challenge.

Additionally, this project aims to democratize music creation by making the process more accessible through AI-driven tools. For individuals without formal musical training, the ability to generate music with minimal input—by simply selecting parameters such as the instrument, number of notes, and temperature—opens up new possibilities for creative expression. This kind of system empowers users to experiment with music generation, fostering creativity in a way that was traditionally limited to trained musicians.

From a technical standpoint, the motivation lies in demonstrating how deep learning models, particularly RNNs, can learn from large datasets of musical sequences and generate new compositions that maintain the inherent structure and flow of music. By investigating the role of RNNs in generating novel music, the project also seeks to further the understanding of AI's capabilities in handling complex, creative tasks, thus contributing to the broader discourse on the role of AI in creative domains.

1.3PROJECT OBJECTIVES

- Develop a Music Generation System using Recurrent Neural Networks (RNNs) that can autonomously compose sequences of musical notes.
- Utilize a dataset of MIDI files to train the RNN model on musical patterns such as pitch, duration, and step size.
- Incorporate an interactive user interface that allows users to control key parameters such as instrument selection, temperature, and the number of notes to be generated.
- Generate coherent and musically meaningful compositions by predicting the next note in a sequence based on learned patterns.
- Evaluate the model's performance through visual and audio outputs, such as piano roll visualizations and playable MIDI files.
- Explore the role of deep learning in creative arts, particularly the application of RNNs to music composition.
- Provide a flexible and user-friendly system for both trained musicians and non-musicians to experiment with AI-driven music creation.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 OVERVIEW OF MUSIC GENERATION

Music generation using artificial intelligence (AI) combines creativity with computational techniques to enable machines to compose music autonomously. By analyzing patterns in existing music, AI models—particularly Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks—are trained to process sequential data and generate new musical compositions. These models learn to predict the next note in a sequence, creating coherent and musically meaningful outputs.

AI-driven music generation has broad applications, from composing original music for films and games to assisting musicians in their creative process. By leveraging AI, music creation becomes more accessible, allowing non-musicians to experiment with composition. This technology not only expands the creative possibilities for professional composers but also introduces a new frontier for collaboration between human creativity and intelligent systems.

2.2 OVERVIEW RECURRENT NEURAL NETWORKS

Recurrent Neural Networks (RNNs) are a type of neural network designed to handle sequential data by maintaining a memory of previous inputs. Unlike traditional neural networks, where inputs and outputs are independent, RNNs allow information to persist across time steps, making them particularly effective for tasks where context is important, such as time series analysis, natural language processing, and music generation. This memory is achieved through feedback loops within the network, enabling RNNs to use prior information to influence current decisions.

The architecture of RNNs is especially suited for tasks involving sequences, as it processes input data step by step while retaining the necessary context from earlier steps. In the context of music generation, RNNs can model the relationships between musical notes, rhythms, and patterns over time, allowing them to predict the next note in a melody based on previous notes. However, standard RNNs can struggle with long-term dependencies, which has led to the development of more advanced variants like Long Short-Term Memory (LSTM) networks that are better at retaining long-range context. These advancements make RNNs a powerful tool for tasks requiring an understanding of temporal relationships.

2.3 LITERATURE SURVEY

S.no	Title of the Paper	Description	Keywords
1	Music Generation Using an LSTM	This paper presents an approach to music generation using LSTMs, detailing how RNNs, particularly LSTMs, can be applied to sequence modeling in music. It provides insights into the challenges of generating structured and melodious outputs.	LSTM, music generation, sequence modeling
2	Differential Music: Automated Music Generation Using LSTM Networks with Representation Based on Melodic and Harmonic Intervals	This paper introduces a novel approach to music generation using LSTM networks, where the focus is on encoding musical information based on melodic and harmonic intervals rather than absolute pitches. This representation offers a unique method of composing music using interval-based encoding and shows promising results	LSTM, melodic intervals, harmonic intervals
3	Automated Music Generation Using Recurrent Neural Networks	This work provides an overview of using RNNs for automated music generation, with a focus on symbolic music. The authors explain the use of various neural network architectures to model musical sequences and generate coherent compositions. Additionally, the paper explores how deep learning models can be trained to create original music compositions based on large datasets	RNN, symbolic music, deep learning
4	MidiNet: A Convolutional GAN for Symbolic-Domain Music Generation	This paper explores the use of Generative Adversarial Networks (GANs) in combination with RNNs for music generation. The researchers demonstrate how combining these models improves the coherence and quality of generated music, particularly in the symbolic domain of MIDI representations	GAN, symbolic music, music generation
5	Chord Generation from Symbolic Melody Using BLSTM Networks	This paper focuses on chord generation using Bidirectional LSTMs (BLSTMs) for symbolic melodies. It investigates how harmonization can be automatically performed and how models can generate chords that match a given melody, providing a broader perspective on how different aspects of music can be generated through RNN-based models	BLSTM, chord generation, music harmonization

CHAPTER 3

TECHNOLOGIES AND TOOLS USED

- **TensorFlow and Keras:** These are the primary frameworks used for building and training the Recurrent Neural Network (RNN) model. TensorFlow, with its Keras API, provides high-level abstractions for defining neural networks, managing datasets, and optimizing models for sequential data tasks like music generation.
- **PrettyMIDI:** This library is used for handling MIDI data, a format that encodes musical information such as pitch, note duration, and velocity. PrettyMIDI enables easy reading, writing, and manipulation of MIDI files, allowing the model to work with musical data.
- **Fluidsynth:** A software synthesizer that is used to convert MIDI files into audio waveforms. This tool plays a key role in generating the actual sound of the music, allowing the generated MIDI notes to be heard as musical output.
- **Ipywidgets:** This tool is used to create an interactive user interface in Jupyter notebooks. Through sliders and dropdowns, users can adjust parameters like the number of notes, temperature, and instrument, providing a customizable experience for generating music.
- **NumPy and Pandas:** These libraries are used for data manipulation and handling the numerical data required for training the model. NumPy is essential for handling large arrays and matrices, while Pandas facilitates structured data processing and analysis.
- **Matplotlib and Seaborn:** These are visualization libraries used to plot the piano roll (a graphical representation of musical sequences) and to display the distributions of generated notes, pitch, duration, and timing.

CHAPTER 4

DATASET

For this project, the **MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization)** dataset is used, which consists of a large collection of classical piano performances in both **MIDI** and **audio** formats. The dataset contains detailed musical information, making it highly suitable for training deep learning models for music generation.

Key Features of the Dataset:

- **Total Size**: Approximately **1282 MIDI files** and corresponding audio recordings, representing over **200 hours** of piano performances.
- **Duration**: Pieces range in length from a few minutes to more extended performances.
- **Composers**: The dataset includes compositions from a wide variety of classical composers, contributing to the richness and diversity of the training data.
- **Years**: The performances span from **2004 to 2018**, providing a contemporary dataset of high-quality classical performances.

MIDI Data Structure:

Each MIDI file in the dataset contains detailed information for every note played, including:

- **Pitch**: The note's frequency (e.g., C4, D5).
- **Velocity**: The intensity with which the note is played, influencing its volume.
- **Start and End Times**: These define the **duration** of the note.
- **Step**: The time interval between successive notes.

Sample Data:

Note	Pitch	Start Time	End Time	Duration	Velocity
1	60 (C4)	1.000s	1.500s	0.500s	64
2	64 (E4)	1.500s	2.000s	0.500s	70
3	67 (G4)	2.000s	2.500s	0.500s	65

In this example:

- **Pitch** represents the frequency of the note (e.g., C4).
- **Start Time** and **End Time** define when the note begins and ends.
- **Duration** is the length of time the note is played.
- **Velocity** controls the volume or intensity of the note, adding dynamics to the music.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

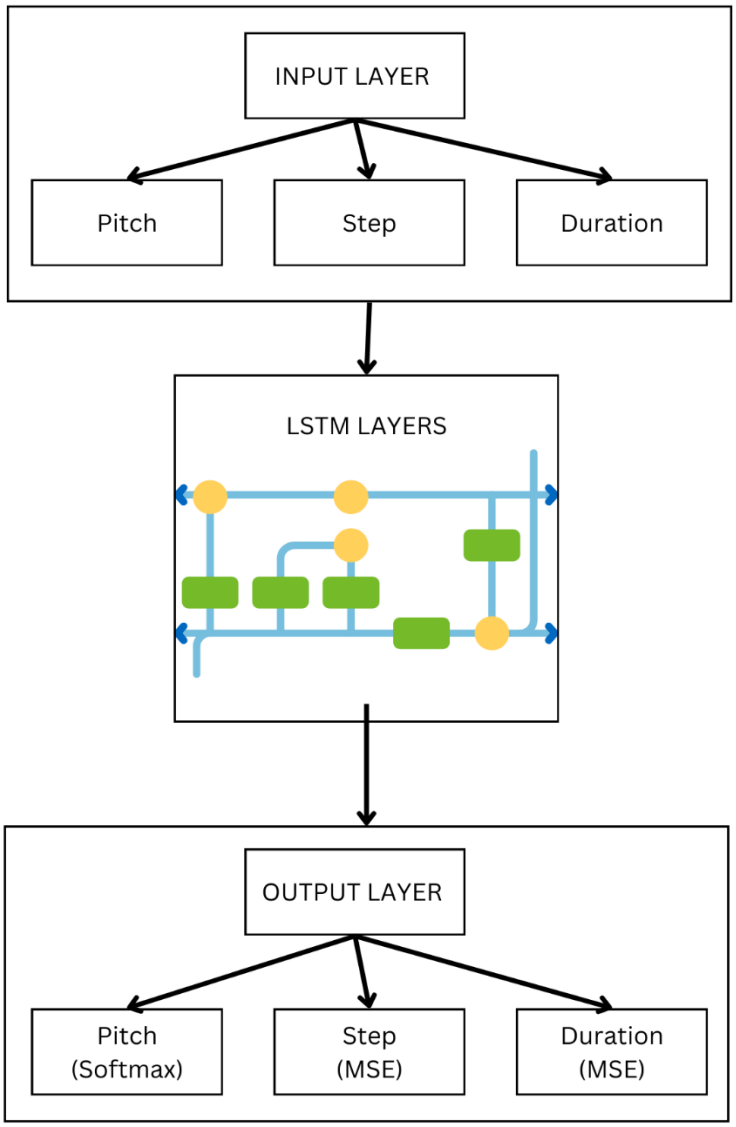


Figure 1 System Architecture

The architecture consists of the following components:

1. **Input Layer**: The model accepts sequences of musical notes, each represented by three features: **Pitch**, **Step**, and **Duration**. These features are fed into the model for each time step in the sequence.
2. **LSTM Layers**: The LSTM layers process the input sequence and capture the temporal dependencies between notes. The recurrent connections within the LSTM allow the model to learn how musical notes relate to one another over time, enabling it to predict future notes based on past sequences.
3. **Output Layer**: The output from the LSTM layers is split into three branches:

- **Pitch Prediction (Softmax)**: This branch predicts the pitch of the next note by generating a probability distribution over 128 possible MIDI note values using a softmax function.
- **Step Prediction (MSE)**: This branch predicts the time interval (step) between the current note and the next, using Mean Squared Error (MSE) for optimization.
- **Duration Prediction (MSE)**: This branch predicts the duration of the next note, also using Mean Squared Error (MSE) as the loss function.

The model is optimized using a combination of categorical cross-entropy for pitch prediction and MSE for both step and duration predictions, ensuring that the generated notes are musically coherent and time-accurate.

5.2 WORKFLOW DIAGRAM

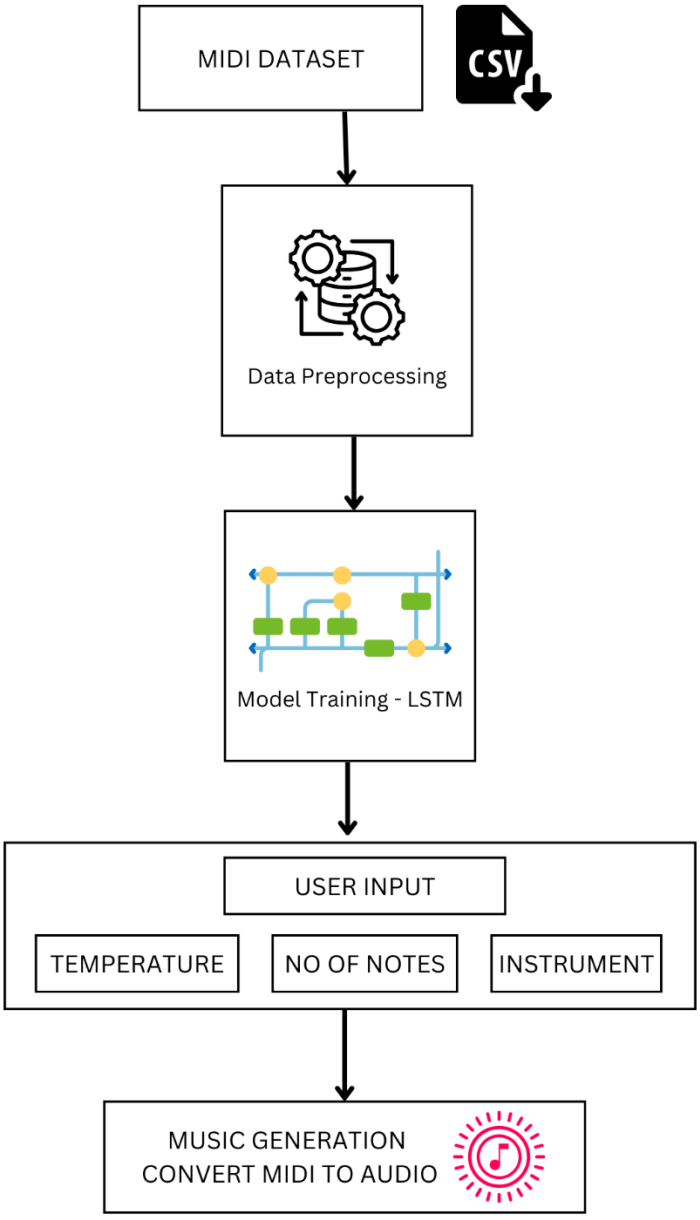


Figure 2 Workflow Diagram

The workflow diagram illustrates the end-to-end process of the **Music Generation System** using a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) layers. The diagram breaks down the key stages involved in generating music based on user input and a trained deep learning model. The major steps are:

1. **MIDI Dataset**: The process begins with the input of MIDI data from the MAESTRO dataset, containing musical information like pitch, step, and duration for each note.
2. **Data Preprocessing**: The raw MIDI data is preprocessed to extract relevant features. This includes converting the MIDI files into structured data containing the pitch, step (time between notes), and duration of each note, preparing it for model training.
3. **Model Training**: The extracted data is used to train an RNN model with LSTM layers, enabling it to learn the temporal dependencies between musical notes. The model learns to predict the next note in a sequence based on past notes.
4. **User Input**: After the model is trained, users can interact with the system by selecting parameters such as **temperature**, **number of notes**, and **instrument** through an interactive interface. This allows for customization of the generated music.
5. **Music Generation**: Based on the trained model and user input, the system generates new sequences of notes by predicting the pitch, step, and duration of the next note in the sequence.
6. **MIDI to Audio Conversion**: The generated note sequences are converted from MIDI format into an audio waveform using **Fluidsynth** or another synthesizer, enabling users to listen to the generated music.
7. **Generated Music Output**: Finally, the system outputs the generated music, allowing users to play back the audio and view visualizations of the musical sequences (e.g., piano roll, note distributions).

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 DEVELOPMENT ENVIRONMENT SETUP

The project was developed using **Python** as the primary programming language, with a suite of libraries and tools to facilitate deep learning, MIDI processing, and user interaction. The system was built and tested in the **Google Colab** environment, which provided access to free computational resources (GPUs) to accelerate the training of the RNN model. The following libraries were used:

- **TensorFlow/Keras**: For building, training, and optimizing the RNN model.
- **PrettyMIDI**: For handling and processing MIDI data.
- **NumPy** and **Pandas**: For data manipulation and processing.
- **Matplotlib** and **Seaborn**: For visualizing the musical data (e.g., piano rolls and distributions).
- **Ipywidgets**: To create interactive controls that allow users to set parameters for music generation, such as the number of notes, temperature, and instrument type.
- **Fluidsynth**: To convert the generated MIDI files into audible audio waveforms.

6.2 DATA PREPROCESSING

The **MAESTRO dataset** was used as the source of musical data. The MIDI files in the dataset were processed to extract important features like **pitch**, **step** (time between successive notes), and **duration** (the length a note is played). These features were essential for training the model.

Steps involved in data preprocessing:

1. **Loading MIDI files**: The system read MIDI files from the MAESTRO dataset using PrettyMIDI.
2. **Feature extraction**: For each note in the MIDI file, the system extracted pitch, step, and duration and stored them in a structured format using Pandas.
3. **Normalization**: The pitch data was normalized, and step and duration were scaled to ensure consistent input ranges for the neural network model.

6.3 MODEL DESIGN AND TRAINING

The core of the system is a **Recurrent Neural Network (RNN)** with **Long Short-Term Memory (LSTM)** layers. The model architecture was designed to predict the next note in a sequence, including its pitch, step, and duration. The key components of the model include:

- **Input Layer**: Accepts sequences of notes where each note is represented by three features: pitch, step, and duration.
- **LSTM Layers**: These layers process the sequence of notes, learning temporal dependencies and patterns in the musical data.
- **Output Layers**: Three separate dense layers output the predictions for pitch (via softmax), step (via MSE), and duration (via MSE).

The model was compiled using the **Adam optimizer** and trained on the preprocessed data. To ensure model efficiency and accuracy, multiple loss functions were used, including categorical cross-entropy for pitch and mean squared error (MSE) for step and duration predictions.

6.4 INTERACTIVE USER INTERFACE

An essential feature of the system is the **interactive user interface** implemented using **Ipywidgets**. This interface allows users to interact with the trained model by adjusting key parameters before generating music. The following interactive controls were implemented:

- **Temperature Slider**: Adjusts the model's temperature, controlling the randomness of the generated notes.
- **Number of Notes Slider**: Allows users to define how many notes the model should generate.
- **Instrument Dropdown**: Lets users select the instrument for which the notes will be generated (e.g., piano, violin, flute).

Users can adjust these settings in real time, providing a flexible and customizable music generation experience.

6.5 MUSIC GENERATION AND AUDIO CONVERSION

Once the model generates a sequence of notes based on the user's input, the system uses **PrettyMIDI** to convert the predicted notes into a **MIDI file**. This MIDI file is then converted into an audio waveform using **Fluidsynth**, enabling users to listen to the generated music directly.

Steps involved:

1. **Note Prediction**: The model generates a sequence of notes by predicting pitch, step, and duration for each note based on the user's input parameters.
2. **MIDI File Creation**: The generated notes are compiled into a MIDI file format.
3. **Audio Conversion**: Using Fluidsynth, the MIDI file is converted into a playable audio file, allowing users to hear the generated music.

6.6 VISUALIZATION

To provide users with a better understanding of the generated music, the system also includes **visualizations**. The following visual outputs were implemented:

- **Piano Roll**: A graphical representation of the generated music that shows how the pitch of each note changes over time.
- **Distributions of Notes**: Histograms showing the distribution of pitch, step, and duration values in the generated sequence.

These visualizations are created using **Matplotlib** and **Seaborn** and are displayed alongside the generated music.

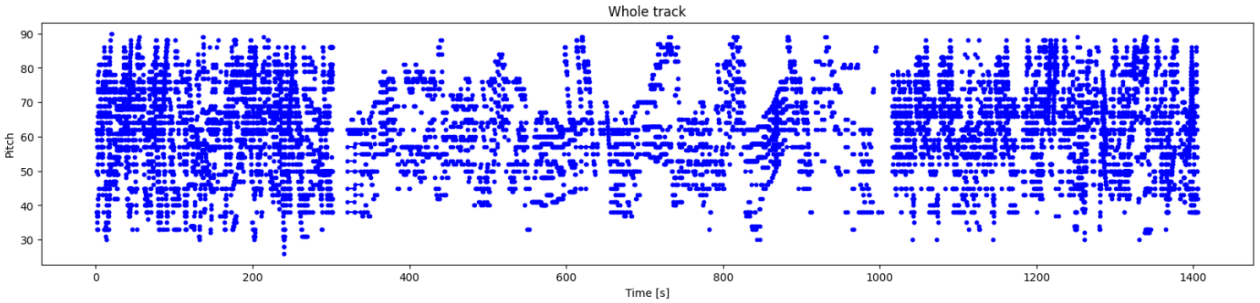
CHAPTER 7

EVALUATION AND RESULTS

7.1 OBSERVATION

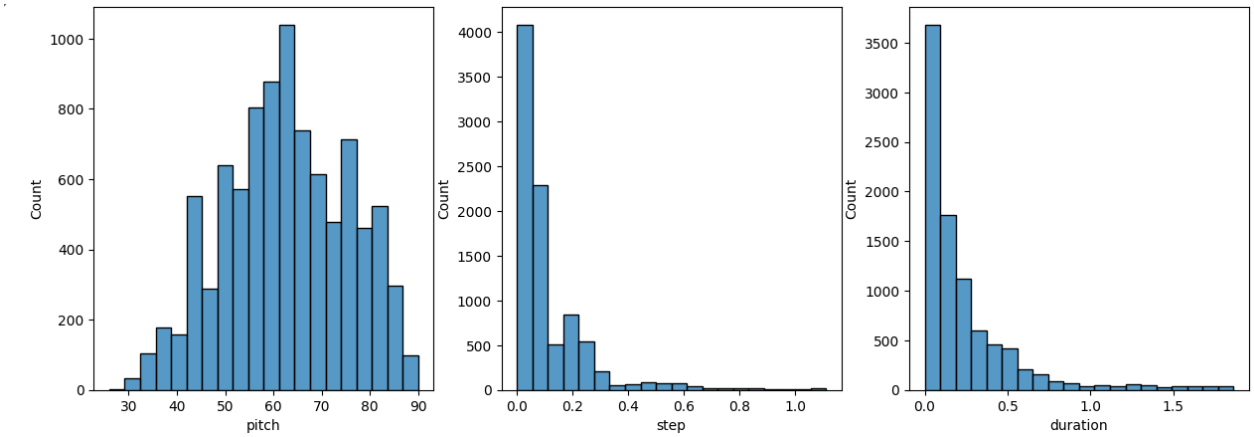
(i) Music notes visualization:

This visualization highlights the coherence of the generated sequence and shows whether the model maintains a smooth musical flow.



(ii) Histograms:

The histograms show the distribution of the generated notes in terms of pitch, step (time between notes), and duration.

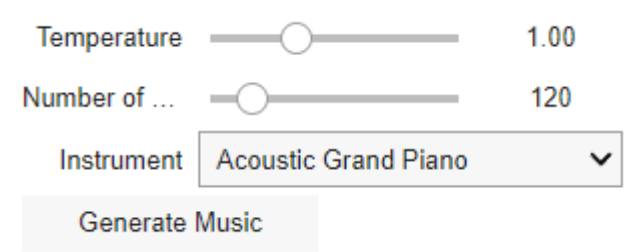


(iii) Audio Outputs:

```
[25] display_audio(example_pm)
```

(iv) Interactive Controls:

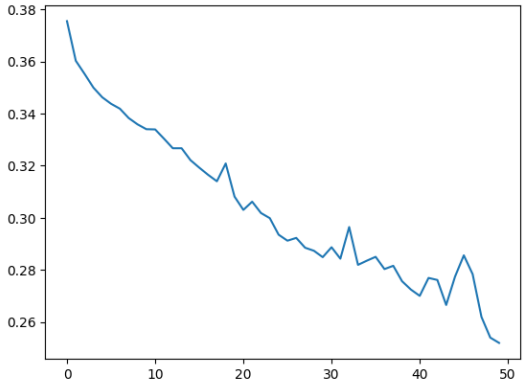
The system allows users to adjust parameters like temperature, number of notes, and instrument type.



7.2 SYSTEM EVALUATION

(i) Training and Validation loss:

This plot shows the **training loss** decreasing over 50 epochs as the model learns from the training data. Below the plot, you have additional metrics from evaluating the model on a validation set.



duration_mean_squared_error: 0.0254 - loss: 0.2094 - pitch_sparse_categorical_accuracy: 0.1123 - step_mean_squared_error: 0.0102 Validation Losses: {'duration_mean_squared_error': 0.01797800324857235, 'loss': 0.20351870357990265, 'pitch_sparse_categorical_accuracy': 0.10183823853731155, 'step_mean_squared_error': 0.009174637496471405}

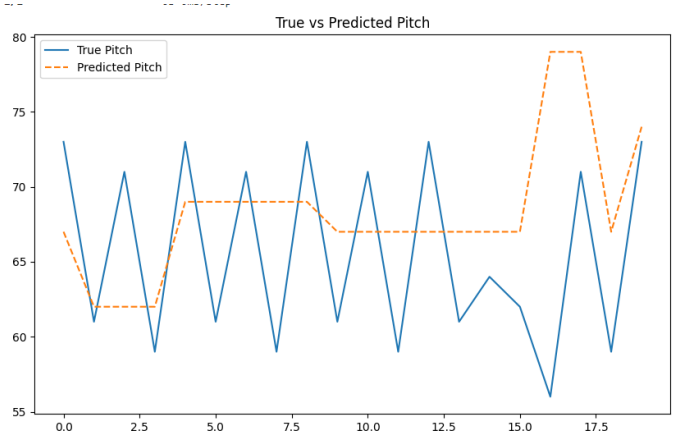
Training Loss: The final training loss after 50 epochs is approximately **0.26**, indicating that the model was able to minimize the prediction error on the training set effectively.

Validation Loss: The validation loss is reported as **0.2035**, which is close to the training loss. This indicates that the model is generalizing well to unseen data and is not overfitting, as the gap between training and validation loss is minimal.

(ii) True Vs Predicted Pitch:

The plot above compares the **true pitch** values from the validation dataset with the **predicted pitch** values from the model for the first 20 notes in a sequence.

- **X-axis**: Represents the index of the notes in the sequence (from 0 to 19).
- **Y-axis**: Represents the **pitch values** of the notes, which correspond to MIDI pitch numbers. Higher values indicate higher pitch notes, while lower values indicate lower pitch notes.



(iii) Model Summary:

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 25, 3)	0	-
lstm (LSTM)	(None, 128)	67,584	input_layer[0][0]
duration (Dense)	(None, 1)	129	lstm[0][0]
pitch (Dense)	(None, 128)	16,512	lstm[0][0]
step (Dense)	(None, 1)	129	lstm[0][0]

Total params: 84,354 (329.51 KB)
Trainable params: 84,354 (329.51 KB)
Non-trainable params: 0 (0.00 B)

The table above provides a summary of the neural network architecture used in the music generation system. The model follows a **functional API** design, consisting of an input layer, a Long Short-Term Memory (LSTM) layer, and three dense layers responsible for predicting pitch, step (time between notes), and duration. The following key components are described in detail:

- **Input Layer**: The input layer accepts sequences of shape (None, 25, 3), where 25 represents the sequence length (number of notes) and 3 represents the features (pitch, step, duration) for each note.
- **LSTM Layer**: The LSTM layer contains 128 units and is responsible for learning the temporal dependencies in the music sequence. It outputs a sequence of size (None, 128).
- **Dense Layers**: There are three dense output layers:
    - **Pitch Prediction Layer**: Outputs a vector of size 128, corresponding to the possible pitch classes. The softmax activation is applied during prediction.
    - **Step and Duration Prediction Layers**: Each of these layers outputs a single value representing the predicted step (time between notes) and duration for the next note.
- **Parameter Count**:
    - The model has a total of **84,354 trainable parameters**.
    - The **LSTM layer** contributes the largest portion of parameters, with 67,584 parameters.
    - Each dense layer contributes additional parameters based on the output size (e.g., the pitch layer has 16,512 parameters).
- **Complexity**: The model has a manageable complexity with 84,354 trainable parameters, making it lightweight compared to more complex deep learning architectures.
- **Parameter Distribution**: Most parameters are concentrated in the LSTM layer, which is responsible for capturing the temporal dynamics of the input music sequences. The dense layers following the LSTM layer are smaller, focusing on individual predictions (pitch, step, and duration).
- **Modularity**: The architecture is modular, with separate output layers for predicting different aspects of the generated music, allowing flexibility in improving or tuning each feature (pitch, step, duration) independently.

CHAPTER 8

CONCLUSION

In this project, the use of Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) units, was explored for the purpose of generating music. The model was trained on sequential musical data with the goal of predicting key musical elements, including pitch, step (time intervals between notes), and duration. A system was developed to allow user interaction, enabling the generation of customized music through adjustable parameters such as temperature, number of notes, and instrument selection.

The results demonstrated that the model performed well in predicting **step** and **duration** with relatively low error. However, the accuracy of **pitch prediction** was found to be limited, indicating that there is significant room for improvement in this area. This suggests that while the model is capable of generating coherent rhythmic patterns, it faces challenges in replicating the complexity and variability inherent in musical pitch sequences. Nevertheless, the system was able to generate basic musical patterns, demonstrating its potential for further development in automated music composition.

Future improvements could focus on enhancing the accuracy of pitch prediction, experimenting with more advanced neural network architectures, and incorporating additional musical features such as chords and dynamics. Furthermore, the system's interactive capabilities offer the potential for developing a more user-friendly tool for music composition. Overall, a solid foundation has been established for continued research and innovation in the field of AI-based music generation.

CHAPTER 9

REFERENCES

[1] Agrawal, Piyush, et al. "Automated Music Generation using LSTM." *International Conference on Computing for Sustainable Global Development*. 2018.

[2] Rafraf, H. (2021). Differential Music: Automated Music Generation Using LSTM Networks with Representation Based on Melodic and Harmonic Intervals. *arXiv preprint arXiv:2108.10449*. Retrieved from [arxiv.org]

[3] Czyz, M., Kedziora, M. (2021). Automated Music Generation Using Recurrent Neural Networks. In: Zamojski, W., Mazurkiewicz, J., Sugier, J., Walkowiak, T., Kacprzyk, J. (eds) Theory and Engineering of Dependable Computer Systems and Networks. DepCoS-RELCOMEX 2021. Advances in Intelligent Systems and Computing, vol 1389. Springer, Cham. https://doi.org/10.1007/978-3-030-76773-0_3

[4] Yang, Li-Chia, Szu-Yu Chou, and Yi-Hsuan Yang. "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation." *arXiv preprint arXiv:1703.10847* (2017).

[5] Lim, Hyungui, Seungyeon Rhyu, and Kyogu Lee. "Chord generation from symbolic melody using BLSTM networks." *arXiv preprint arXiv:1712.01011* (2017).

CHAPTER 10
APPENDIX

Code:

Github Link:
https://github.com/SanthoshiRavi/Music-Generator-Using-LSTM.git

Google Colab Notebook link:
https://colab.research.google.com/drive/1h9vjaJ61pg0ALNNWYGSki9i_rrBXY4XV?usp=sharing