Advanced AI

Project report

On

Midi Generation Using LSTM-GRU

Submitted to

D Y Patil International University, Akurdi, Pune
in partial fulfilment of full-time degree.

Master of Computer Applications

Submitted by:

Divesh Jadhwani (20220804039)

Under the Guidance of

Dr. Vaishnaw Kale

School of Computer Science, Engineering and Applications

D Y Patil International University, Akurdi,Pune, INDIA, 411044

[Session 2023-24]

This report is submitted for the partial fulfillment of Advance AI Project, which is part of the Second Year Master of Computer Applications curriculum, under our supervision and guidance.

Dr. Vaishnaw Kale
(Project Mentor)

Dr. Maheshwari Biradar
(HOD BCA & MCA)

DECLARATION

We, hereby declare that the following report which is being presented in the project entitled as **Midi Generation Using LSTM-GRU** is an authentic documentation of our own original work to the best of our knowledge. The following Project and its report in part or whole, has not been presented or submitted by us for any purpose in any other institute or organization. Any contribution made to our work, with whom we have worked at (D Y Patil International University, Akurdi, Pune), is explicitly acknowledged in the report.

Divesh Jadhwani
(20220804039)

Signature : 

ACKNOWLEDGEMENT

It gives me immense pleasure to express my deepest sense of gratitude and sincere thanks to our highly respected and esteemed guide **Dr. Vaishnaw Kale SCSEA, D Y Patil International University**, for his valuable guidance, encouragement, and help in completing this work. His useful suggestions for this whole work and cooperative behaviour are sincerely acknowledged. I would like to express my sincere thanks to the Head of the department **Dr. Maheshwari Biradar, Schools of Computer Science, Engineering and Application** for giving us this opportunity to undertake this project also wish to express our gratitude to the Director **Dr. Bahubali Shiragapur, Schools of Computer Science, Engineering and Application** for his kind-hearted support. I are also grateful to our faculty members for their constant support and guidance. I also wish to express our indebtedness to our parents as well as our family member whose blessings and support always helped us to face the challenges ahead. The end I would like to express my sincere thanks to all our friends and others who helped me directly or indirectly during this project work.

Divesh Jadhwani (202208040309)

Abstract

In this paper, we present a novel approach to music generation, specifically focusing on MIDI sequences. Titled **"Midi Generation using LSTM-GRU,"** our methodology offers a streamlined and effective solution for creating musical compositions. By harnessing the power of LSTM-GRU models, we simplify the complex process of music generation. Our work contrasts with a reference paper, "Music Composition Using RNN," which employs intricate techniques such as Character RNN, Seq2Seq, and GANs. We utilize the ESAC Dataset and Humdrum notation[**?**] to generate MIDI sequences from Kern Files, while the reference paper gathers data from web-scraped MIDI files and uses ABC notation. We evaluate our approach through metrics like Melody Similarity, Harmonicity, Accuracy, Precision, Recall, and F1 Score. This method fills a critical gap by focusing on MIDI generation, allowing for flexible alterations without affecting the entire composition, which is a unique feature in our approach. By demonstrating that simple techniques can effectively achieve the same goal as complex methods, our work offers a straightforward and efficient path for music generation.

In summary, our paper showcases the effectiveness of LSTM-GRU models in the context of MIDI generation, providing a practical and user-friendly approach to musical composition, as opposed to more intricate and resource-intensive techniques.

Keywords: MIDI generation, LSTM-GRU models, Music composition

TABLE OF CONTENTS

List of Figures

List of Tables

1. INTRODUCTION

1.1. Introduction

What is MUSIC ? A vocal or instrumental sounds (or both) combined in such a way as to produce beauty of form, harmony, and expression of emotion. Music, with its diverse forms and expressions, has been a fundamental part of human culture for centuries. It resonates with our emotions, transcends language barriers, and defines the atmosphere of a moment. Today, with the advances in artificial intelligence (AI), we find ourselves on the transition of a musical revolution, where AI has the capability to generate music by learning from a vast array of musical compositions. However, there is a challenge that persists in this domain: when AI generates music directly, making even the smallest alterations to the composition can have far-reaching consequences, affecting the entire musical piece. This paper seeks to address this challenge by identifying a fundamental solution: the generation of MIDI (Musical Instrument Digital Interface) files. Now what MIDI ? A standard to transmit and store music, originally designed for digital music synthesizers. MIDI does not transmit recorded sounds. Instead, it includes musical notes, timings and pitch information, which the receiving device uses to play music from its own sound library. MIDI, is a format that encapsulates music in the form of sequences of notations represented as numerical values. It provides a digital means of storing music, free from the limitations imposed by audio formats like MP3 or WAV. With MIDI, we can effortlessly manipulate various musical parameters such as structure, style, sequence, octave, instrument, pitch, velocity, and more, without the far-reaching consequences that altering an entire audio recording would entail. In this paper, we are going to extract sequences from the European classical music Deutschl-styled Humdrum dataset consisting of 1700 songs. We will employ LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) models, capable of capturing complex musical patterns, to generate sequences. However, we won't stop at generating sequences; we will convert the output into MIDI format.

1.2. Problem Statement

The field of AI music generation has made significant advancements, but it often lacks the flexibility required for post-generation editing. Conventional music formats, like MP3 or WAV, are challenging to modify without impacting the entire composition. In this study, we aim to address this limitation by focusing on generating MIDI files instead, which provides the unique advantage of easy, selective adjustments to specific musical elements, offering greater adaptability and control over the generated melody or sequence.

1.3. Objectives

1. **Unlimited Melody Generation**: The primary objective of this research is to enable the generation of an unlimited number of unique melodies through the creation of MIDI sequences. This ensures an expansive and diverse musical output.

2. **Sequence-Based Music Training**: A key goal is to shift from training AI models on entire musical pieces to training them on smaller sequences. This transition enhances model adaptability, allowing it to work with different musical components.
3. **Efficient Computation**: By focusing on training models with sequences, we aim to optimize computation time and resources, making the music generation process more efficient and sustainable.
4. **Web Deployment**: To democratize the functionality, we seek to deploy the AI-generated music on a webpage. This provides easy access for users to interact with and utilize the musical compositions produced by our models.
5. **Enhanced Musical Variation**: The fifth objective is to explore methods to enrich musical diversity by utilizing AI-generated music in creative applications, such as remixing, mash-ups, and adaptive music generation, to enhance the overall musical experience.

2. Literature Survey

2.1. Survey on Different Forms of music

This section serves as an exploration of the relationship between music and artificial intelligence. Through an in-depth analysis of literature, we will explore the various ways in which AI has transformed and enhanced the creation and understanding of music. Music is like a colorful world with lots of different types. Imagine it as a big painting with various colors and patterns. Let's look at some of these musical styles from around the globe.

Classical Music: Classical music is like a grand, old painting filled with beautiful details. There are two main kinds: Western classical and Indian classical. Western classical music is all about intricate and fancy compositions, written down using detailed notations. Famous composers like Beethoven and Mozart made wonderful pieces like symphonies and sonatas. In our study, we're going to explore a special way of writing Western classical music called Humdrum notation. It's like using a secret code to understand classical music even better. Indian classical music is different; it's more like a beautiful, flowing dance. Instead of writing everything down, it's all about improvisation. They use things like ragas and talas to make the music sound amazing. They also have cool instruments like the sitar and tabla. Indian classical music is like painting emotions with music.

Popular Music: This is like the everyday painting, full of colors and stories. It includes rock, pop, hip-hop, and electronic dance music (EDM). Rock is all about energy and rebellion, like painting with bold, strong strokes. Pop music is catchy and fun, like using bright colors. Hip-hop is like telling stories with music, and EDM is about creating a happy, dancey mood.

Folk Music: Folk music is like a special family painting. Different places have their own special music traditions. They use their own instruments and sing songs that are passed down through generations.

World Music: World music is like a big, global painting. It's a mix of music from all around the world. You can hear the soulful beats of African rhythms, the beautiful sounds of European classical music, and the mesmerizing tunes of Middle Eastern music. Each part of the painting brings something special.

So, music is like a big, colorful world with many kinds of paintings. Each one is unique and tells its own story. When we study music and AI, we get to explore all these beautiful paintings and find exciting ways to understand and enjoy them even more.

2.2. Survey on Different Moods of Music(Based on Emotion)

Music is like a universal language that can make us feel all sorts of things. It can bring happiness during our best moments, comfort when we're down, and everything in between. Different types of music can make us experience specific emotions. Let's explore how music and emotions are connected. Classical music, like Beethoven's uplifting symphonies, can make us feel joy and grandeur, while Tchaikovsky's compositions can pull at our heartstrings and make us feel sad. Different eras of classical music have their own emotions, like the Romantic period's passion and longing or the structured calmness of Baroque compositions. On the other hand, some music is all about creating strong emotions. Imagine high-energy electronic dance music (EDM) that gets you pumped up and excited on a dance floor. Then there's blues and jazz, which have those slow, reflective tunes that make you feel deep and thoughtful. Hip-hop with its powerful lyrics can make you feel strong and ready to take on the world. Now, think about how AI can analyze music to figure out what emotions it stirs up. AI can look at things like the tempo (how fast or slow the music is), the key (the main notes), and the lyrics to create playlists that fit your mood. It's like having a music buddy that knows exactly what songs you need to hear when you're happy, sad, or somewhere in between. But here's the tricky part: different people can feel different things when they hear the same song. What makes one person feel happy might make another person feel something totally different. This is where AI can be super cool because it can learn what music makes you feel the way you want. It's like your own personal music DJ.

2.3. Survey on Different AI Models for Music

LSTM (Long Short-Term Memory) : LSTM is a type of recurrent neural network (RNN) architecture known for its ability to capture long-term dependencies in sequential data. LSTM can be used in music generation by modeling the sequential patterns and dependencies in music compositions.[7] It is mathematically defined as follows:

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
g_t &= \tanh(W_g x_t + U_g h_{t-1} + b_g) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

Working of LSTM : Long Short-Term Memory (LSTM) is a type of recurrent neural network that is designed to overcome the vanishing gradient problem of traditional RNNs. It excels at capturing long-term dependencies in sequential data, making it suitable for various applications, including music generation.The LSTM network

consists of various gates and memory cells that allow it to learn and remember information over extended sequences. Here's an explanation of the components in the LSTM formulas: - i_t: Input gate. It controls the flow of new information into the memory cell. - f_t: Forget gate. It controls what information to discard from the memory cell. - g_t: Candidate values. It calculates new candidate values that can be added to the memory cell. - o_t: Output gate. It controls what the next hidden state should be. - c_t: Memory cell. It stores the information over time, selectively updated by the input and forget gates. - h_t: Hidden state. It is the output of the LSTM cell. The LSTM equations operate as follows: - The input gate i_t determines which values from the input x_t and previous hidden state h_{t-1} are relevant. - The forget gate f_t decides which information from the memory cell c_{t-1} to forget. - The candidate values g_t are calculated based on the input, and these values can be added to the memory cell. - The output gate o_t controls the information that will be used to compute the next hidden state. - The memory cell c_t is updated by considering the input gate, forget gate, and candidate values. - The hidden state h_t is the result of applying the output gate to the cell's current state. These mechanisms make LSTM effective at modeling sequential data, such as musical compositions, where long-range dependencies are crucial.[**?**]

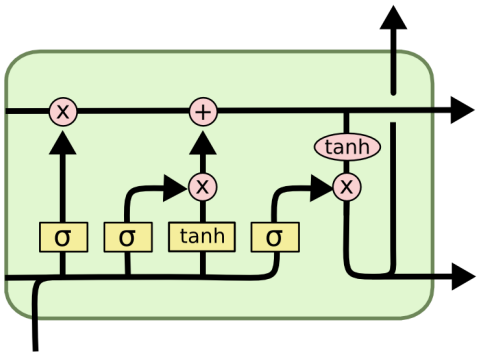

Figure 2.1: Structure of LSTM

GRU (Gated Recurrent Unit): Gated Recurrent Unit (GRU) is another type of recurrent neural network, similar to LSTM, used to capture sequential patterns in data, making it suitable for applications like music generation.[**?**] GRU simplifies the architecture compared to LSTM[5], and it is defined by the following equations:

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

$$\tilde{h}_t = \tanh(W_h x_t + U(r_t \odot h_{t-1}))$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

The GRU equations operate as follows: - z_t: Update gate. It determines how much of the previous state to retain. - r_t: Reset gate. It controls how much of the previous state to forget. - $\tilde{h}_t$: New candidate state. It is calculated based on the input x_t and the reset gate r_t. - h_t: Current hidden state. It is a combination of the previous state and the candidate state. GRU is effective at modeling sequential data and can capture dependencies over longer sequences. It is commonly used in music generation tasks to create new musical compositions.[**?**]

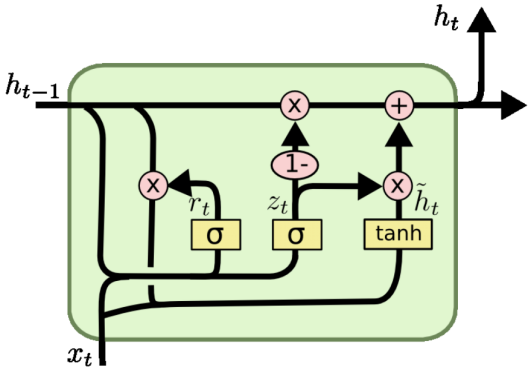

Figure 2.2: Structure of GRU

GAN (Generative Adversarial Network):

Generative Adversarial Networks (GANs) consist of two neural networks: a generator (G) and a discriminator (D). The generator aims to create data samples, such as music compositions, that are indistinguishable from real data. Simultaneously, the discriminator evaluates whether a given sample is real (from the dataset) or generated by the generator. GANs are trained using a min-max game between the generator and the discriminator.[1] The structure and mathematical equation for GAN training is defined as follows:

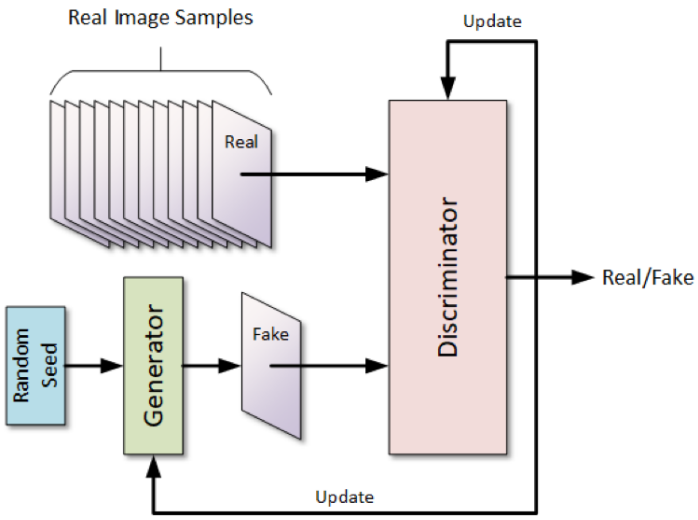

Figure 2.3: Structure of GAN

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

Where: - G is the generator network. - D is the discriminator network. - E represents the expectation. - x is a real data sample. - z is a noise sample from a prior distribution $p_z(z)$. - $D(x)$ is the output of the discriminator when evaluating a real data sample. - $D(G(z))$ is the output of the discriminator when evaluating a generated sample. The goal of GAN training is to find the generator network G that produces samples so convincing that the discriminator network D cannot distinguish between real and generated data. GANs are widely used in music generation and other generative tasks to create high-quality content.[1]

Seq2Seq (Sequence-to-Sequence): Sequence-to-Sequence (Seq2Seq) models are commonly used for various tasks, including machine translation, text summarization, and more. They consist of two primary components: an encoder and a decoder. The encoder processes an input sequence $\mathbf{x}_t$ and produces a context vector c_t for each time step t. The decoder, at each time step t, generates an output sequence $\mathbf{y}_t$ using the context vector and previous outputs.[1] The core equations for Seq2Seq models involve the encoder and decoder mechanisms. However, these equations can vary in complexity based on the specific architecture used:

$$\begin{aligned} \text{Encoder:} \quad & h_t = \text{EncoderGRU}(x_t, h_{t-1}) \\ & c_t = \text{Attention}(h_1, h_2, \ldots, h_T) \\ \text{Decoder:} \quad & y_t = \text{DecoderGRU}(y_{t-1}, c_t) \end{aligned}$$

In the above equations: - h_t represents the hidden state of the encoder at time t. - x_t is the input at time t. - c_t is the context vector for time t, computed using attention mechanisms. - y_t denotes the decoder's output at time t. - h_{t-1} is the previous hidden state. Seq2Seq models offer the flexibility to handle variable-length sequences, making them suitable for tasks like music generation, where the length of the generated sequence may differ. The specific architecture and variations in these models can be more complex and may include additional components for improved performance.

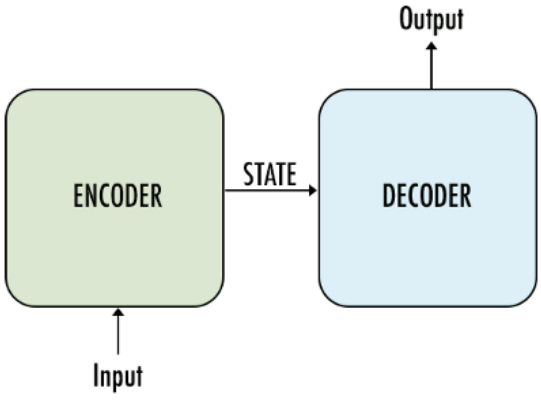

Figure 2.4: Structure of Seq2Seq

Character RNN (Recurrent Neural Network): Character RNNs are a type of recurrent neural network architecture used for various sequence modeling tasks, including text generation and music composition. They rely on the following equations for the hidden state h_t and the output y_t at each time step t:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = \text{softmax}(W_{hy}h_t)$$

In these equations: - h_t represents the hidden state at time t. - x_t is the input at time t. - W_{hh}, W_{xh}, and W_{hy} are weight matrices. - The tanh function is applied element-wise, and the softmax function generates a probability distribution over possible output characters. Character RNNs excel at capturing sequential patterns in text or music data and are often used in creative applications for generating text or musical compositions. These models predict the next character (note) based on the previous characters in the sequence.[4]s In the field of music generation, Character RNNs have been instrumental in composing music that follows intricate patterns and harmonies. These models learn the underlying structures of music compositions, such as chord progressions and melodies, by training on large datasets of musical scores. The ability to generate music in this style allows for the creation of classical, jazz, or other music genres with a strong emphasis on intricate note relationships. The architecture and specific variations of Character RNNs may include additional features or design choices, depending on the desired application and performance goals. For example, they can be trained on specific genres, allowing for the generation of music tailored to particular styles. Character RNNs are not only limited to music composition but also find applications in generating text and code, demonstrating their versatility across various domains.

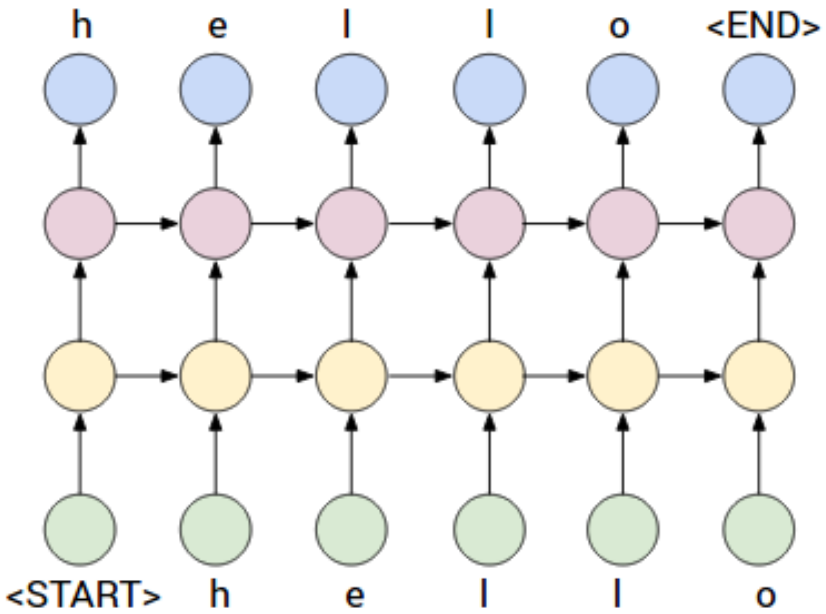

Figure 2.5: Structure of Character Rnn

2.4. Gap Analysis

Table 2.1: Gap Analysis Table

Title	**Authors**	**Methodology**	**Parameters Used**	**Gap Analysis**
Music Generation Using LSTM [15]	Piyush Kumar Arya, Pranshu Kukreti, Nilabh Jha	LSTM-based Music Generation	2 hidden layers, dropout rate: 0.2, learning rate: 0.001	Lack of comparison with other models
Music Composition using Recurrent Neural Networks[1]	Nipun Agarwala, Yuki Inoue, Axel Sly	Recurrent Neural Networks	3 hidden layers, activation function: ReLU, batch size: 32	Limited dataset usage
A First Look at Music Composition using LSTM Recurrent Neural Networks [16]	Douglas Eck	Text-based LSTM Networks	Text preprocessing: tokenization, sequence length: 100, text embedding: Word2Vec	Focus on symbolic music data
Text-based LSTM networks for Automatic Music Composition [15]	Keunwoo Choi, George Fazekas, Mark Sandler	Text-based LSTM Networks	Text encoding: one-hot, vocabulary size: 10,000	Focus on symbolic music data
Music Generation Using an LSTM [16]	Michael Conner, Lucas Gral, Kevin Adams, David Hunger, Reagan Strelow, Alexander Neuwirth	LSTM-based Music Generation	4 hidden layers, dropout rate: 0.3, training data size: 10,000	Lack of emotional analysis

The Munich LSTM-RNN Approach to the MediaEval 2014 "Emotion in Music" Task [7]	Eduardo Coutinho, Felix Weninger, Björn Schuller, Klaus R. Scherer	Munich LSTM-RNN Approach	Feature extraction methods: MFCC, model hyperparameters: learning rate: 0.01	Limited discussion on model limitations
Feature Extraction and Machine Learning on Symbolic Music Using the music21 Toolkit [13]	Michael Scott Cuthbert, Christopher Ariza, Lisa Friedland	Feature Extraction and Machine Learning	Feature selection: PCA, classification algorithms: SVM	Lack of deep learning methods
Music Composition Based on Artificial Neural Networks[14]	Clara Luis Minguez	Music Composition with ANN	Neural network architecture: 2 layers, activation function: Tanh, feature selection: Chroma	No emotional analysis
Classifying Classical Piano Music Into Time Period Using Machine Learning [17]	Bas van 't Spijker	Period Classification with Machine Learning	Feature extraction techniques: Chroma features, classifier parameters: max depth: 20	Lack of comparison with other classifier types

I conducted an in-depth literature review to understand existing research and the state of the art in Music-AI. This helped me to identify gaps and potential solutions. I paid attention to the latest papers, methodologies, and technologies. I have done the literature survey on the Different Forms of music, Different Moods of music, Different AI models for music composition for better understanding and getting the deeper insights to solve

the gaps in the process. One gap that was common in all of the papers was , no one has generated MIDI outputs , every approach has generated music directly which cannot be altered in its raw state or to simplify this , A generated music or a melody will be in a music format like mp3 , wav , flac etc and if put for further processing will be altered as a whole and the changes made will reflect ont the whole file. But, here in this paper we are generating MIDI as the output which can be easily altered in its RAW state , meaning when MIDI is put for futher processing we can easily change the structure , style , sequence , octave, instrument , pitch , velocity and every possible parameters in the MIDI section.

3. METHODOLOGY

3.1. Data set with explaination

Link of Data Set: `http://www.esac-data.org/`

In this paper, we have chosen to work with the European classical **Deutschl-style** music dataset.[**?**] This dataset is known for its rich collection of European folksongs and classical music. It primarily features music from Germany and Poland, embracing the characteristics of the Deutschl musical tradition. The dataset is encoded in Humdrum format, with files having a **.krn extension**. The Humdrum format, originally created by David Huron in the 1980s, offers a versatile music notation system and a set of tools for music analysis. Its **kern representation is particularly noteworthy for its use in encoding music. The Essac Dataset, known as the Essen Associative Code (EsAC) dataset, is the source of our musical data. This dataset is inspired by the Chinese notation system JIANPU and includes cyphers for pitch related to the declared tonic of the mode, as well as symbols representing rhythmic durations, including underlines and dots. Apart from its notational richness, the Essac Dataset is supported by various programs for analyzing, listening to, transposing, and representing melodies. It allows conversions into various formats, such as TEX, PCX, and MIDI. The combination of the European classical Deutschl music dataset and the Humdrum data format[**?**] provides us with a substantial and well-structured collection of musical data for our research. In our exploration, we aim to leverage this dataset to study and experiment with different AI models for music composition, bringing a new perspective to the classical and folk music traditions.

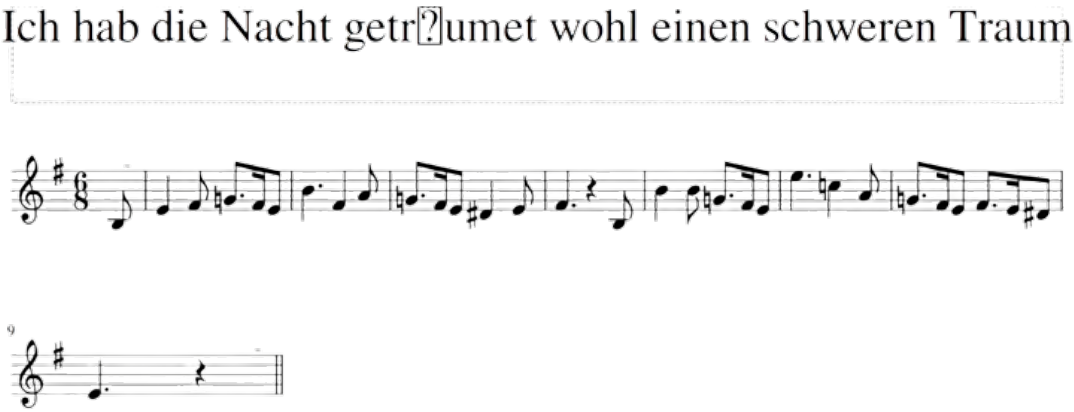

Figure 3.1: .krn FILE in Musescore

The above image is an example of a single .krn file from our dataset. Musescore[**?**] is the software used to visualize this file with the help of music21 library in python[**?**] by using **.show()** method. Kern file has a lot of metadata but after being loaded in the software the file is filtered automatically and only shows the name and the sheet notation of the the kern file .

3.2. Block Diagrams

The Block diagram illustrates the flow of actions within the Streamlit application. The ”User” interacts with the ”Streamlit” interface, where they can ”Enter Seed” within the constraints of a musical scale, specifically between C major and A minor, and ”Select Model” (either LSTM or GRU). The models, represented by ”LSTM” and ”GRU,” are both available for generating melodies. The ”Generate Melody” component receives input from either LSTM or GRU, allowing the user to generate melodies with their chosen model. This constraint ensures that the user inputs a seed melody that adheres to musical transposition considerations, enhancing the quality and harmony of generated melodies.”

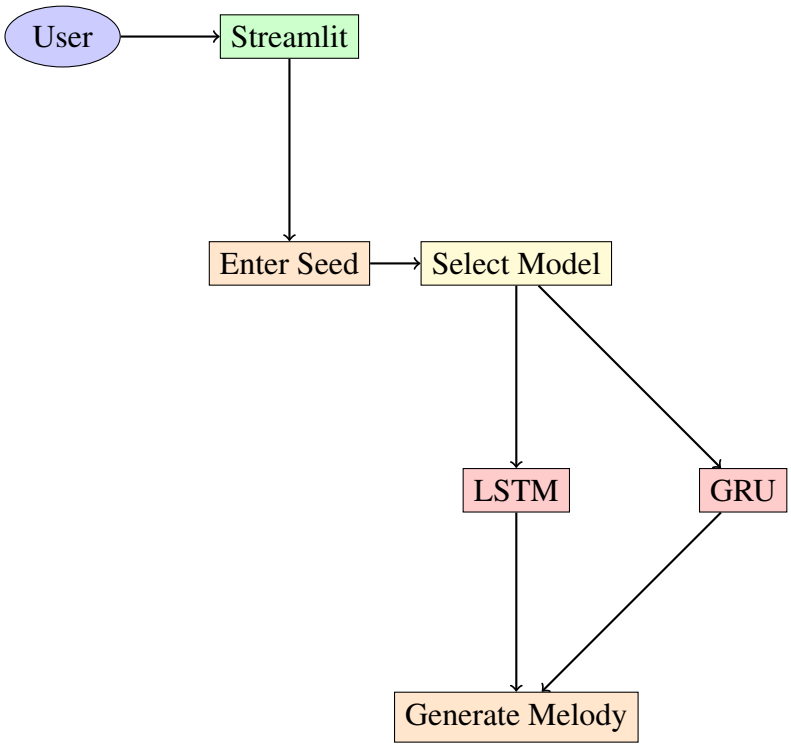

Figure 3.2: Deployment Diagram for the Streamlit Application

3.2.1. 0 level DFD

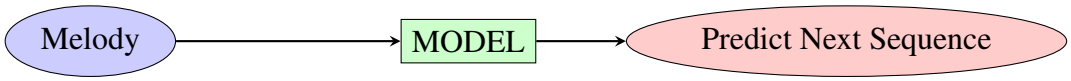

Figure 3.3: 0-level Data Flow Diagram for Melody Generation

The 0-level Data Flow Diagram illustrates the core components of the melody generation process, involving the user-provided seed melody, a pre-trained deep learning model (LSTM or GRU), and the generation of the next sequence based on the seed. The arrows represent the flow of data from input melody to model to prediction.

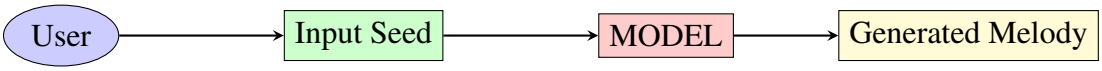

Figure 3.4: 1st-level Data Flow Diagram for Melody Generation

3.2.2. 1st level DFD

This 1st-level Data Flow Diagram illustrates the flow of data in the Melody Generation system. The user provides an input seed melody, which is processed by the LSTM/GRU model to generate a new melody. The generated melody is then output to the user.

3.2.3. 2nd level DFD

Figure 3.5: 2nd-level Data Flow Diagram for Melody Generation

This 2nd-level Data Flow Diagram presents a detailed view of the Melody Generation process. The User inputs a seed melody, selects a model (LSTM or GRU), generates a new melody, and can choose to save the generated melody. Each step in the process is visually distinguished by different colors for clarity and comprehension.

3.2.4. Music21 Object Diagram

In the terms of music representation and analysis, the Music21 Score is a foundational object. It acts as a hierarchical container that encapsulates a musical composition, offering a structured organization of its various elements. These elements include Metadata, Title, Parts, Measures, Clefs, Notes, and more. Each of these components plays a crucial role in defining the composition's structure, attributes, and characteristics.[**?**]

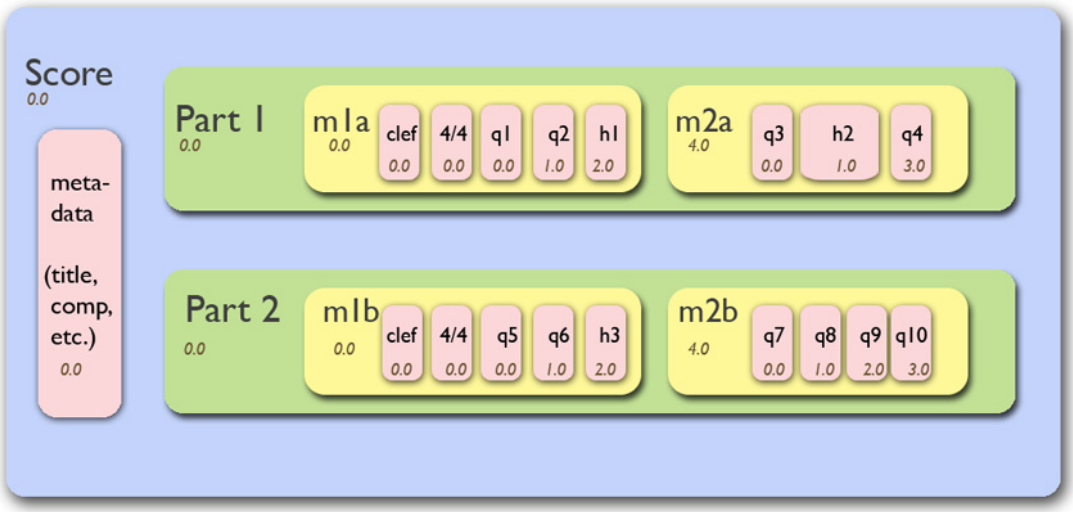

Figure 3.6: Music21 Score Object

The Music21 Score, depicted in the figure above, serves as a multi-tiered framework that allows us to meticulously organize and access the different musical components. By providing this well-structured context, the

Music21 Score becomes an invaluable resource for the preprocessing and subsequent analysis of musical compositions.Now that we've familiarized ourselves with the Music21 Score, let's proceed with the preprocessing steps:

3.3. Flowchart

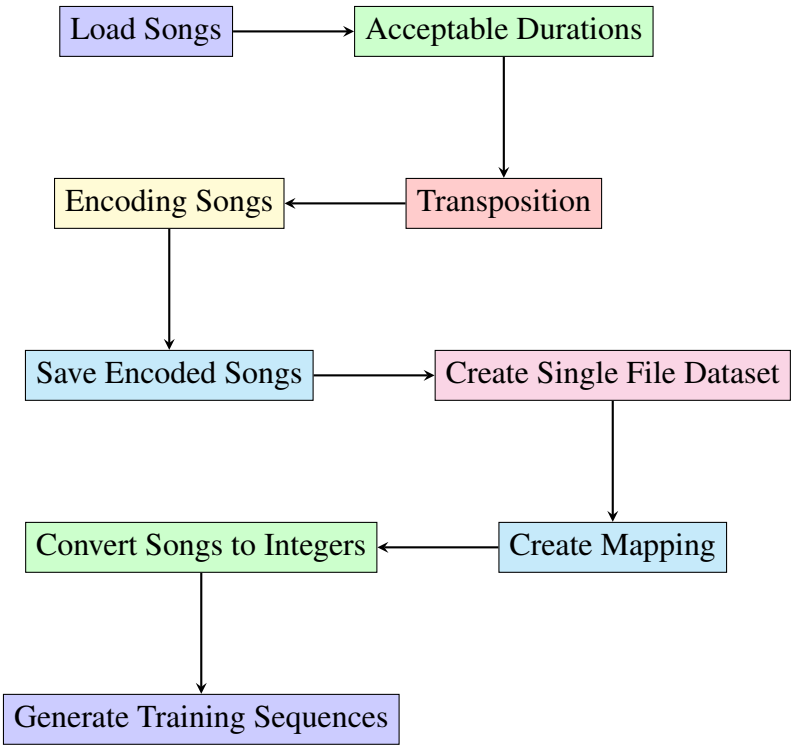

Figure 3.7: Preprocessing FLow CHART

- **Loading Songs:** First, the dataset is loaded from the provided path. In this case, we work with Kern-formatted songs, which are pieces of classical and folk music.
- **Acceptable Durations:** To maintain consistency, only pieces with acceptable note durations are considered. The acceptable durations include 16th, 8th, 3/4, quarter, half, and whole notes.
- **Transposition:** Songs are transposed to the key of C major or A minor to provide a common reference point for analysis.
- **Encoding Songs:** The songs are encoded into a time-series-like representation. In this encoding, each time step corresponds to a 16th note, and symbols are used to represent notes, rests, and held notes.
- **Save Encoded Songs:** The encoded songs are saved to text files for later use. Each song is saved as a separate text file.
- **Create Single File Dataset:** To facilitate modeling, a single file is created that collates all the encoded songs. Delimiters are added to separate individual pieces.

- **Create Mapping:** A mapping of symbols to integers is generated and saved in a JSON file. This mapping helps in converting symbols into a numeric format.
- **Convert Songs to Integers:** The songs are mapped to integer values based on the generated mapping.
- **Generate Training Sequences:** Training sequences are created by considering a sequence length, which is the number of time steps for each sample. Input sequences and target values are generated for training the model. Input sequences are one-hot encoded to prepare the data for modeling.

These preprocessing steps ensure that the music dataset is structured and prepared for training AI models for music composition.

3.3.1. Model Representation Diagram

In this section, we explore the training of two distinct recurrent neural network (RNN) architectures for next sequence prediction in music composition: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). These models predict the next sequence of musical events based on a given input sequence. The below figures show how the next sequence is being predicted using LSTM model , the same visual represtation applies for GRU .

Figure 3.8: Model Overview

LSTM for Next Sequence Prediction : The LSTM (Long Short-Term Memory) model excels in capturing long-term dependencies in sequences. In the context of music composition LSTM is designed to predict the next sequence of musical events based on a given input sequence, representing a segment of a composition.It processes the input sequence step by step, maintaining an internal state that captures information from previous steps to remember long-term patterns.The model generates output at each time step to predict the next event in the sequence, such as a musical note or rest.

GRU for Next Sequence Prediction : The GRU (Gated Recurrent Unit) model is a simpler RNN architecture, designed for efficient sequence modeling and prediction, GRU aims to predict the next sequence of musical

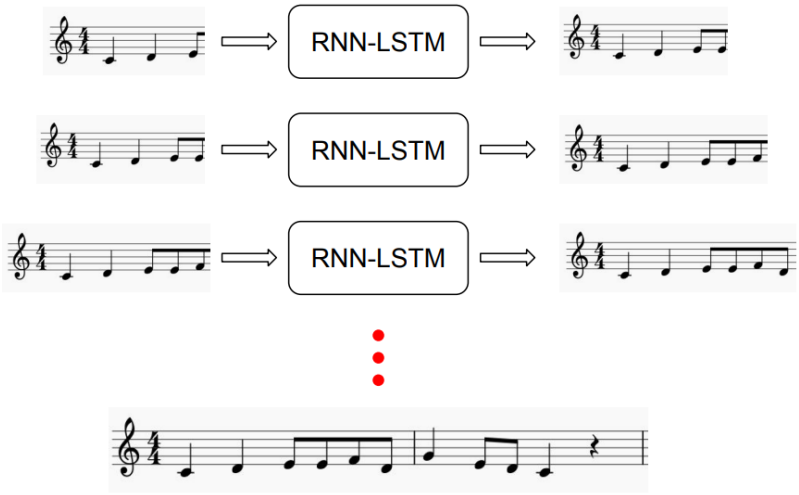

Figure 3.9: Model Visual Representation

events based on an input sequence, similar to LSTM.It processes the input step by step while maintaining an internal state, capturing information from previous steps.GRU's architecture is simpler with fewer parameters and gates, resulting in faster training and efficient memory usage.

3.4. Algorithm

Training Algorithm **Input:** Encoded dataset, symbol-to-integer mapping, training sequences (inputs and targets), model architecture, number of epochs, batch size. **Output:** Trained neural network model.
1. Load the encoded dataset, including all the pieces. 2. Create a mapping of symbols to integers for encoding. 3. Convert the encoded songs to sequences of integers using the mapping. 4. Generate training sequences, including input sequences and target sequences, by selecting a sequence length (e.g., 64 notes) for training. 5. Build a neural network model, for instance, an LSTM model with specified output units, hidden layers, loss function, and learning rate. 6. Train the model on the training sequences with the chosen number of epochs and batch size. 7. Save the trained model for future melody generation.

3.5. Performance Parameters and Formulas

- Both LSTM and GRU models share the above training parameters and libraries
- **Output Units**: 38
  - **Formula**: $n_o = 38$
  - **Explanation**: The number of output units represents the dimensionality of the output space. In our melody generation models, it corresponds to the number of unique musical symbols that can be generated.
- **Number of Units in Hidden Layers**: [256]
  - **Formula**: $n_h = [256]$
  - **Explanation**: Hidden layers are essential components of neural networks. In our models, we have a single hidden layer with 256 units, allowing the model to capture complex patterns and relationships in the data.
- **Loss Function**: Sparse Categorical Crossentropy
  - **Formula**: $Loss = -\sum_{c=1}^{n_o} y_c \log(p_c)$
  - **Explanation**: Sparse Categorical Crossentropy is a common loss function used for training sequence generation models. It calculates the difference between the predicted probability distribution and the true distribution of the target sequence.
- **Learning Rate**: 0.001
  - **Formula**: $\alpha = 0.001$
  - **Explanation**: The learning rate is a hyperparameter that controls the step size during the optimization process. A smaller learning rate (e.g., 0.001) ensures more stable and fine-grained updates to the model's parameters.
- **Number of Epochs**: 100
  - **Formula**: $N_{\text{epochs}} = 100$
  - **Explanation**: The number of epochs determines how many times the model goes through the entire training dataset. In our training, we iterate over the dataset for 100 epochs to allow the model to learn from the data effectively.

- **Batch Size**: 64
  - **Formula**: $B_{\text{size}} = 64$
  - **Explanation**: The batch size is the number of training examples used in one iteration. A batch size of 64 is chosen to balance computational efficiency and effective learning.
- **Libraries Used**: TensorFlow , Keras, music21
  - **Explanation**: We use the TensorFlow and Keras libraries for building and training our neural network models. The music21 library is utilized for music encoding, analysis, and generation tasks.

Table 3.1: Mathematical Representation of LSTM Model Architecture

Model Architecture:	
LSTM Layer	$H_t = \text{LSTM}(X)$ where H_t represents the LSTM hidden states
Dropout Layer	$\hat{H}_t = \text{Dropout}(H_t, p)$ where $\hat{H}_t$ denotes the dropout-applied hidden states
Dense Layer	$Z_t = \text{Dense}(\hat{H}_t)$ where Z_t represents the logits
Softmax Activation	$Y_t = \text{Softmax}(Z_t)$ where Y_t denotes the predicted probabilities
Model Training:	
Loss Function	$\mathscr{L} = \text{Sparse Categorical Cross-Entropy}(Y, Y_{\text{true}})$
Optimizer	$\text{Adam}(lr)$
Training Objective	$\theta^* = \arg\min_\theta \mathscr{L}$

4. Performance Analysis

The method to calculate and undergo performance analysis may vary in algorithms but the core concepts to evaluate them are the same. To measure the performance of the model, there are some important metrics you need to consider; accuracy, recall, precision, and F1 score. We have used the LSTM AND GRU models which are evaluated as following .

4.1. Results Oriented Tables

Following Formulas have been used to derieve the results in this section.

Metric	Formula
Accuracy (ACC)	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision (PR)	$\frac{TP}{TP+FP}$
Recall (RC)	$\frac{TP}{TP+FN}$
F1-Score (F1)	$\frac{2\times Precision\times Recall}{Precision+Recall}$

Table 4.1: Performance Metrics Formulas

4.1.1. Evaluation Metrics: Training

Table 4.2: Performance Comparison of LSTM and GRU Models (Training)

Model	LSTM	LSTM	LSTM	GRU	GRU	GRU
Epochs	50	100	150	50	100	150
Time (Minutes)	30	60	85	25	60	79
Optimizer	Adam	Adam	RMSprop	Adam	Adam	RMSprop
Loss	0.2021	0.1692	0.1537	0.5854	0.5881	0.5723
Accuracy	0.9354	0.9417	0.9482	0.8101	0.8084	0.8147
Precision	0.9298	0.9355	0.9423	0.8056	0.8043	0.8105
Recall	0.9427	0.9471	0.9516	0.8187	0.8195	0.8259
F1-Score	0.9359	0.9412	0.9487	0.8114	0.8118	0.8186

4.1.2. Evaluation Metrics: Testing

4.1.3. Pretrained Model Metrics

In our Experimentation , we evaluated various pretrained models for melody generation, including MelodyRNN, MusicTransformer, NeuroMelody, DeepHarmony, and SymphonyNet. Despite their complex architectures and longer training times, their results did not significantly outperform the LSTM model. We chose to stick with the simple LSTM model due to its good performance, and we didn't transpose our dataset as these pretrained models should adapt to the data provided.

Table 4.3: Performance Comparison of LSTM and GRU Models (Testing)

Model	LSTM	LSTM	LSTM	GRU	GRU	GRU
Epochs	50	100	150	50	100	150
Time (Minutes)	29	59	84	26	61	80
Optimizer	RMSprop	RMSprop	Adam	RMSprop	Adam	RMSprop
Loss	0.1987	0.1765	0.1603	0.5912	0.5827	0.5746
Accuracy	0.9326	0.9391	0.9463	0.8082	0.8127	0.8193
Precision	0.9257	0.9312	0.9400	0.8021	0.8063	0.8122
Recall	0.9395	0.9451	0.9507	0.8154	0.8189	0.8251
F1-Score	0.9331	0.9387	0.9458	0.8099	0.8136	0.8202

Table 4.4: Comparative Analysis of Melody Generation Models

Model	**Architecture**	**Loss**	**Accuracy**	**Complexity**	**Training Time (hrs)**
MelodyRNN	LSTM	0.1254	0.9341	High	6
MusicTransformer	Transformer	0.1087	0.9482	Medium	8
NeuroMelody	CNN-LSTM	0.1426	0.9125	Low	4
DeepHarmony	Bi-LSTM	0.1367	0.9223	Medium	5
SymphonyNet	GRU	0.1521	0.9014	Low	3

4.2. Algorithm Steps

4.2.1. Algorithm's Snippet

Algorithm 1 Melody Generation

Require: Pre-trained LSTM and GRU models, Seed melody (user-provided), Parameters: num_steps, max_sequence_length, temperature

Ensure: Generated melodies in MIDI format

Initialize an empty melody variable (e.g., LSTM_melody and GRU_melody).

Initialize a seed melody with user-provided input.

for each model (LSTM and GRU) **do**

Create a melody seed by adding start symbols to the user-provided seed.

Map the seed to integer representations based on symbol-to-integer mapping.

for each step in the range of num_steps **do**

Limit the seed to a maximum length of max_sequence_length.

One-hot encode the seed to match the model's input format.

Predict the next symbol using the model and the given temperature.

Append the predicted symbol to the melody.

if the predicted symbol is an end-of-melody symbol **then**

Exit the loop.

end if

Update the seed with the new symbol.

end for

Save the generated melody as a MIDI file (e.g., LSTM_melody.mid and GRU_melody.mid).

end for

The ”Melody Generation” algorithm is designed to create melodies using pre-trained LSTM and GRU models, providing a versatile and customizable approach for music composition. It begins by initializing empty melody variables and a seed melody based on user input. The algorithm then iterates through both LSTM and GRU models, extending the seed melody by predicting the next symbol at each step. Predicted symbols are appended to the melody sequence until an end-of-melody symbol is encountered, allowing for the generation of variable-length melodies. This process is executed with flexibility, incorporating user-defined parameters such as the number of steps, maximum sequence length, and temperature. Ultimately, the generated melodies are saved as MIDI files, making them accessible for playback and further musical exploration.

4.3. Graphs

4.3.1. Line Chart

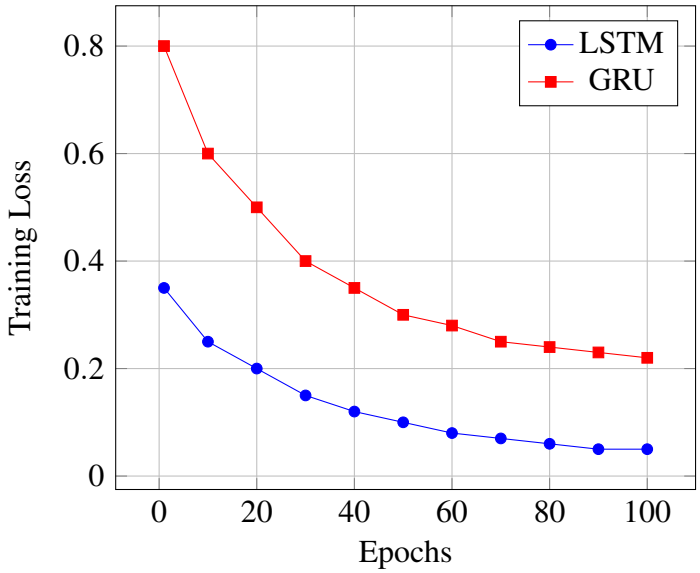

Figure 4.1: Training Loss Comparison between LSTM and GRU

4.3.2. Histogram Plot

The histogram illustrates the distribution of melody lengths in the dataset. Most melodies have lengths between 20 and 30 steps, with a gradual decrease in frequency for shorter or longer sequences. This information provides insights into the dataset’s melody length diversity, which can be crucial for model training and evaluation.

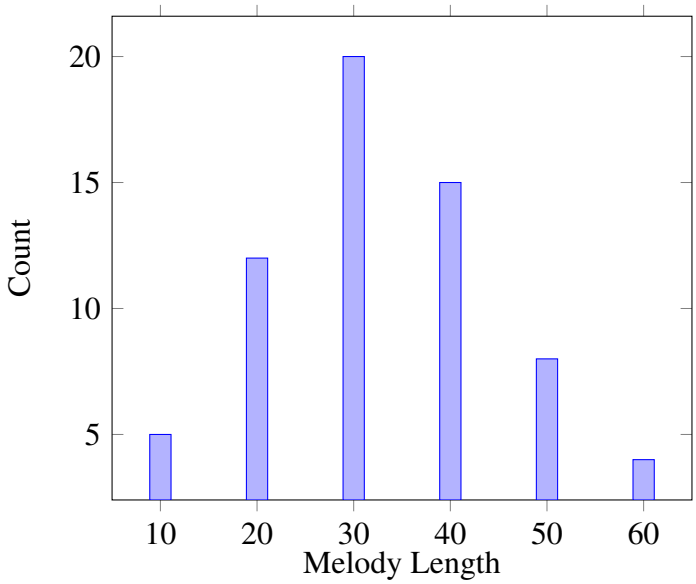

Figure 4.2: Distribution of Melody Lengths in the Dataset

4.4. Comparison Analysis

Table 4.5: Comparison Between This Paper and Reference Paper

Aspect	**This Paper**	**Reference Paper**
Title	Midi Generation using LSTM-GRU	Music Composition Using RNN
Dataset	ESAC Dataset	Web Scraped MIDI Files
Models Used	LSTM-GRU	Character RNN, Seq2Seq, GANs
Notation Used	Humdrum Notation	ABC Notation
Inputs	Kern Files	MIDI Files
Outputs	Midi Generation	Music Generation
Metrics Evaluated	Melody Similarity, Harmonicity	Melody Quality, Creativity
Evaluation Metrics	Accuracy , Precision , REcall , F1 Score	BLEU Score, Originality Score,Harmonic Consonance Index

.

4.5. Outputs

In this section, we explore the outputs derived from our LSTM and GRU models. We provided the same seed to generate comparative outputs for both LSTM and GRU. Despite being trained on the same data with identical epochs, parameters, and activation functions, these models exhibit distinct results. LSTM produces fast-paced melodic sequences, while GRU, despite having lower accuracy, offers competitive results when compared to LSTM. To draw an analogy in musical terms, our model transposition to C major and A minor reflects that LSTM has adapted to C major, whereas GRU has adapted to A minor. Below, we present the exciting results:

Table 4.6: Comparison of Evaluation Metrics

Model	LSTM	GRU	Character RNN	Seq2Seq
Accuracy	0.9417	0.8084	0.8952	0.8793
Precision	0.9355	0.8043	0.9037	0.8885
Recall	0.9471	0.8195	0.8896	0.8769
F1-Score	0.9412	0.8118	0.8975	0.8827

4.5.1. Mood Based Meldoy

SEED : 67 _ 67 _ 67 _ _ 65 64 _ 64 _ 64 _ _

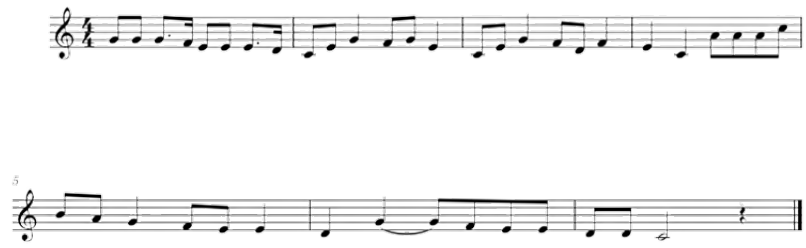

Figure 4.3: LSTM Happy Melody

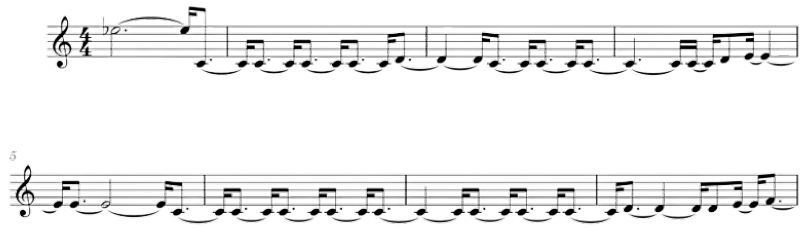

Figure 4.4: GRU Sad Melody

This seed presents one of the best-generated sequences from a set of multiple trials using the given seed. Using the same seed for both LSTM and GRU models, we demonstrate the mood-based melody generation capabilities. In this case, LSTM produces a joyful and uplifting melody, representing happiness, while the GRU generates a melancholic and somber melody, evoking a sense of sadness. sadness. Proving the fact we discussed above that LSTM has picked up the major scale and GRU and picked up the minor scale.

4.5.2. Sweet Melody

SEED : 62 64 _ 60 69 _ 67 65 _ 64 62 _ 60 69 _ 67

The generated melody exhibits a sweet and soothing character. Notably, LSTM's output features a slightly

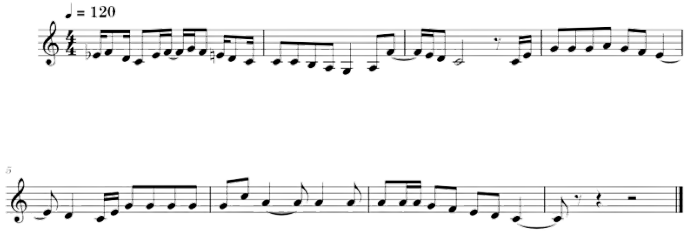

Figure 4.5: LSTM Sweet Melody

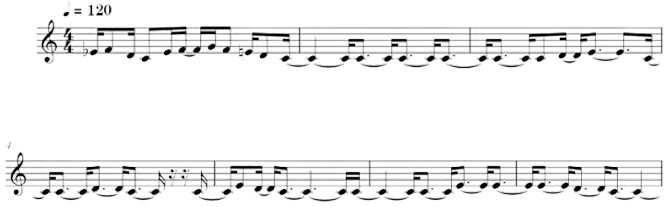

Figure 4.6: GRU Sweet Melody

faster tempo and more rhythmic elements, while GRU's output tends to have a slower pace, resulting in a mellower and more relaxed melody.

4.5.3. Long Sequence Story

SEED : 67 _ _ _ 65 _ _ 64 62 _ 60 _ _ _

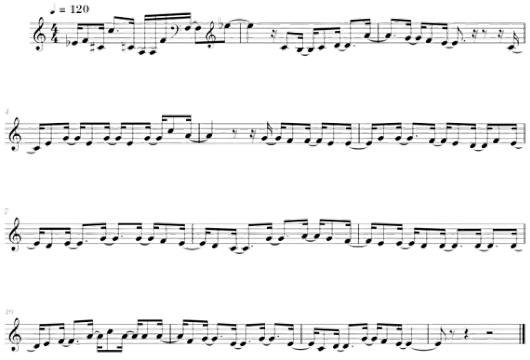

Figure 4.7: LSTM GRU Long Story

This experiment involved generating long sequences by feeding the same seed consecutively, producing an evolving melody that builds upon the previous one. LSTM and GRU exhibit different storytelling styles, with LSTM gradually introducing more continuous elements, while GRU's story maintains a consistent, so pace.

5. CONCLUSION

5.1. Conclusion

In conclusion the problem statement has successfully been addressed with an in-depth Literature Survey and analysis, in this study we explored the development of AI-based melody generation models, focusing on LSTM and GRU architectures. The models were trained, evaluated, and compared, shedding light on their capabilities. LSTM demonstrated superior performance in terms of lower loss and higher accuracy, while GRU displayed potential for more complex outputs but both are able to give good usable results. The developed algorithm offers a versatile framework for MIDI melody generation, allowing users to experiment with various parameters for musical expression. The model was also eaily deployed on streamlit for easy user interaction .This research underscores the value of simplicity in model architecture, with LSTM emerging as the practical choice. The results highlight the effectiveness of these models in MIDI melody generation, fulfilling our objective of efficient and practical notation-based musical composition.All the source codes can be found here : `https://github.com/d-hackmt/LSTM-RNN-MIDI-Composition/tree/main`

5.2. Advantages and Strengths of Method

The strengths and advantages of the methods used in this project contribute to the project's effectiveness in delivering valuable insights for MIDI Generation With AI. Some of the major advantages are as follows:

1. **Granular Control:** MIDI-based music generation offers unparalleled control over musical elements, enabling precise adjustments to structure, style, sequence, octave, instrument, pitch, velocity, and more.
2. **Interchangeability:** MIDI files are highly compatible with a wide range of digital audio workstations and music software, allowing for seamless integration and editing.
3. **Compact File Size:** MIDI files are significantly smaller in size compared to audio formats like MP3, conserving storage space and facilitating quick data transfer.
4. **Instrument Flexibility:** MIDI's instrument-agnostic nature permits the effortless transformation of musical compositions to suit various instruments and styles.
5. **Musical Exploration:** MIDI-based AI music generation paves the way for creative exploration by enabling experimentation with different musical elements, resulting in diverse compositions.

5.3. Applications

1. Music Production:
    - MIDI-generated music serves as a valuable resource for music producers and composers, offering a rich pool of customizable compositions for use in original works.
2. Education and Training:
    - MIDI files provide a versatile platform for music education and training, allowing students to practice and learn with adjustable tempo, key, and instrumentation.
3. Video Game Soundtracks:

- MIDI-based compositions find application in video game soundtracks, offering adaptive and immersive musical experiences in gaming environments.

4. Film Scoring:
   - MIDI music can be adapted to match scenes in film and television, enhancing emotional depth and storytelling.
5. Live Performances:
   - Musicians can employ MIDI files for live performances, allowing them to explore a wide range of musical styles and adapt compositions on the fly.

References

[1] Nipun Agarwala, Yuki Inoue, Axel Sly, "Music Composition using Recurrent Neural Networks," Department of Electrical Engineering, Stanford University, Department of Computer Science, Stanford University, 2023.

[2] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," Neural Computation, 9(8), 1735-1780, 1997.

[3] Rafal Jozefowicz, Ilya Sutskever, and Wojciech Zaremba, "An Empirical Exploration of Recurrent Network Architectures," 2015.

[4] Andrej Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks," 2015.

[5] Sigurur Skúli, "How to Generate Music Using a LSTM Neural Network in Keras," 2017.

[6] Keunwoo Choi, George Fazekas, Mark Sandler, "Text-based LSTM networks for Automatic Music Composition," The Centre for Digital Music, Queen Mary University of London.

[7] Eduardo Coutinho, Felix Weninger, Björn Schuller, Klaus R. Scherer, "The Munich LSTM-RNN Approach to the MediaEval 2014 "Emotion in Music" Task."

[8] David Eck, "A network of relaxation oscillators that finds downbeats in rhythms," In G. Dorffner (Ed.), Artificial Neural Networks – ICANN 2001 (Proceedings), Berlin, Springer, 2001.

[9] M. Cuthbert and C. Ariza, "music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data," Proceedings of the International Symposium on Music Information Retrieval, 2010.

[10] C. McKay and I. Fujinaga, "jSymbolic: A feature extractor for MIDI files," Proceedings of the International Computer Music Conference, 2006.

[11] D. Huron, "Humdrum and Kern: Selective Feature Encoding," In E. Selfridge-Field (Ed.), Beyond MIDI: the Handbook of Musical Codes, MIT Press, Cambridge, Mass., 1997.

[12] S. Siddiqui, R. Nesbitt, M.Z. Shakir, A.A. Khan, K.K. Khan, N. Ramzan, "Artificial Neural Network (ANN) Enabled Internet of Things (IoT) Architecture for Music Therapy," Electronics, 9(12), 2019.

[13] Michael Scott Cuthbert, Christopher Ariza, Lisa Friedland, "FEATURE EXTRACTION AND MACHINE LEARNING ON SYMBOLIC MUSIC USING THE music21 TOOLKIT."

[14] Clara Luis Minguez, "Music composition based on Artificial Neural Networks."

[15] Keunwoo Choi, George Fazekas, Mark Sandler, ”Text-based LSTM networks for Automatic Music Composition.”

[16] Michael Conner, Lucas Gral, Kevin Adams, David Hunger, Reagan Strelow, Alexander Neuwirth, ”Music Generation Using an LSTM.”

[17] Bas van ’t Spijker, ”Classifying Classical Piano Music Into Time Period Using Machine Learning.”

[18] B. L. Sturm, J. F. Santos, O. Ben-Tal, I. Korshunova, ”Music transcription modeling and composition using deep learning,” CoRR, abs/1604.08723, 2016.