



**TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS**

**A
MAJOR PROJECT REPORT
ON
AUTOMATIC MUSIC GENERATION**

SUBMITTED BY:

PRAVESH KHANAL (075BEI026)

SANJEEV BHANDARI (075BEI033)

SRIJANA LAMICHHANE (075BEI038)

SUDIP TAMANG (075BEI041)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

MAY, 2023

Page of Approval

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled "AUTOMATIC MUSIC GENERATION" submitted by **Sanjeev Bhandari, Pravesh Khanal, Sudip Tamang, Srijana Lamichhane** in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

.....
Supervisor
Jyoti Tandukar, Ph.D
Associate Professor
Department of Electronics and Computer
Engineering,
Pulchowk Campus, IOE, TU.

.....
Internal examiner
Person B
Assistant Professor
Department of Electronics and Computer
Engineering,
Pulchowk Campus, IOE, TU.

.....
External examiner
Person C
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

Date of approval:

Copyright

The author has agreed that the Library, Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purposes may be granted by the supervisors who supervised the project work recorded herein or, in their absence, by the Head of the Department wherein the project report was done. It is understood that recognition will be given to the author of this report and the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering, for any use of the material of this project report. Copying or publication, or the other use of this report for financial gain without the approval of the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering, and the author's written permission is prohibited. Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

Head
Department of Electronics and Computer Engineering
Pulchowk Campus, Institute of Engineering, TU
Lalitpur, Nepal.

Acknowledgement

We express our sincere gratitude to the Department of Electronics and Computer Engineering, Central Campus, Pulchowk, for allowing us to choose the major project per the fourth-year syllabus at IOE, Tribhuvan University.

We want to express our deepest gratitude to our supervisor **Assoc. Prof. Jyoti Tandukar, Ph.D.**, for his continuous guidance, direction, and supervision. He has been exceptionally motivating and helpful to us while giving guidance, suggestion, and pointing out the optimal path, which helped us progress our project rapidly.

We want to extend our genuine appreciation to **Asst. Prof. Sanjivan Satyal** for his invaluable guidance and unwavering support during the entire course of our project.

We thank all faculty teachers who provided us with the knowledge to build a foundation, which helped us study the feasibility of our major project and ignited our passion for doing this great project. We would also like to thank our parents and friends, who've been exceptionally motivating and accompanying us throughout our work on the project.

Gratitude,

Pravesh Khanal (075BEI026)

Sanjeev Bhandari (075BEI033)

Srijana Lamichhane (075BEI038)

Sudip Tamang (075BEI041)

Abstract

‘**Automatic Music Generation**’ composes short pieces of music using different parameters like notes, pitch interval, and chords. This project uses the concept of RNN (Recurrent Neural Network) and LSTM (Long Short Term Memory) to generate music using models. The traditional way of composing music requires much trial and error. With automatic music generation, we can predict suitable follow-up music using AI rather than testing music in a studio, effectively saving time.

The main focus of this project is to use the LSTM-NN model and algorithm approach to generate music while ensuring that the output is synchronized between two separate outputs. The dataset used in this project was sourced from the ESAC Folk database[1]. The original format of the dataset was in .kern file format, which was converted into MIDI format for use in this project. MIDI files were used as a music data source, encoded into a time-series notation format, and used to train the model. This project uses the concept of dependencies on the time series music sequence to train a model. The trained model can generate time series notation and decode our generated music to obtain a music file.

Furthermore, we used the concept of tone matrix for the algorithmic approach to generate music that accepts the image as input and creates music based on the image. The image is resized, and a parameter is given to the tone matrix according to pixel intensity. This music can be synchronized with music generated by the AI model to get consistent and pleasing music.

Keywords: *music, midi, lstm, generation, model, notes, tone-matrix*

Contents

Page of Approval	ii
Copyright	iii
Abstract	ii
Contents	v
List of Figures	vii
List of Abbreviations	viii
1 Introduction	1
1.1 Background	1
1.2 Problem Statement	3
1.3 Objectives	3
1.4 Scope	4
2 Literature Review	5
2.1 Magenta	7
2.2 Deep Jazz	8
2.3 Aiva	8

2.4	OpenAI Jukebox	9
2.5	MusicBERT	10
2.6	More Researches	11
3	Related Theory	15
3.1	MIDI	15
3.2	Music Theory	15
3.2.1	Notes	15
3.2.2	Chords	16
3.2.3	Time Signature	17
3.2.4	BPM	17
3.2.5	Sheet Music	17
3.3	Music21	18
3.4	Tone Matrix	18
3.5	RNN and LSTM	20
3.5.1	Forget Gate	21
3.5.2	Input Gate	22
3.5.3	Cell State	22
3.5.4	Output Gate	23
3.5.5	Activation function	23
3.6	Optimizer	24
3.7	Loss	25
3.7.1	Regularization	26

4	Methodology	27
4.1	Block Diagram of Basic Model	27
4.1.1	Data acquisition	28
4.1.2	Data pre-processing	28
4.1.3	Feature extraction	28
4.1.4	Model training	30
4.1.5	Music generation	33
4.1.6	Evaluation	33
4.2	Algorithmic Approach	33
4.3	AI Sync Algorithm	35
4.3.1	Working Mechanism	36
4.4	UML Diagram	37
4.4.1	Use Case Diagram	37
4.4.2	Activity Diagram	38
5	Experiment and Result	39
6	Output Demo	44
7	Conclusion	49
	References	50

List of Figures

3.1	Notes and rests symbol[2]	18
3.2	Example of Tone Matrix with ON (x) and OFF cells	19
3.3	RNN	20
3.4	LSTM	21
4.1	Basic Block Diagram of AI Model	27
4.2	Original file waveform of duration 2 sec	29
4.3	Reconstructed file waveform of duration 2 sec	29
4.4	Model	31
4.5	Training loss	32
4.6	Block Diagram for music generation using algorithmic approach	34
4.7	Block Diagram For AI sync Algorithm Approach	36
4.8	Use Case Diagram	37
4.9	Activity Diagram	38
5.1	Decoding seed in sheet-music ¹	39
5.2	Generated music ¹	40
5.3	Image used for algorithmic approach music generation	40
5.4	Switch configuration using maximum intensity pixel	41
5.5	Sheet music generated by switch of fig (5.4)	41

5.6	Switch configuration using minimum intensity pixel	42
5.7	Sheet music generated by switch of fig (5.6)	42
5.8	Sheet music of multi-instrument music	42
5.9	Sheet music for AI generation	43
5.10	Sheet music for AI sync Algorithm	43
6.1	AI Music Generation Page	44
6.2	Instrument Selection	44
6.3	AI Music Generation Output Page	45
6.4	Sheet Music for AI-Generated Music	45
6.5	Algorithmic Music Generation Page	46
6.6	Algorithmic Music Generation Output Page	46
6.7	Sheet Music for Algorithmic Generated Music	47
6.8	AI Sync Algorithm Music Generation Page	47
6.9	AI sync Algorithm Music Generation Output Page	48
6.10	Sheet Music for AI sync Algorithm Generated Music	48

List of Abbreviations

BPM	Beats Per Minute
CTRL	Creator Technology Research Lab
CUDA	Compute Unified Device Architecture
ESAC	Essen Associative Code
GAN	Generative Adversarial Network
LSTM	Long Short Term Memory
MIDI	Musical Instrument Digital Interface
NLP	Natural Language Processing
NN	Neural Network
PPQ	Pulses Per Quarter note
QPM	Quarter note Per Minute
RNN	Recurrent Neural Network

1. Introduction

Music has always been integral to human culture and can evoke strong emotions and memories. With the advancements in technology, it is now possible to use machine learning techniques to generate original pieces of music. In this project, we implement an AI music generation system using Python and machine learning, which allows users to generate new music by specifying the musical instruments.

We will describe the implementation of our AI music generation system in detail, including the machine learning models and techniques used, the dataset and preprocessing steps, generation of music, and evaluation of the generated music. We will also incorporate an algorithmic approach to generate music based on input images.

We plan to use a real music dataset to train our machine-learning model. Real music can provide a more diverse and representative dataset for training, as it contains a wide range of musical styles and genres. This can help the models learn to generate more realistic music. This project aims to contribute to the growing body of research on AI music generation and demonstrate the feasibility of using machine learning to generate creative music.

1.1 Background

Music has always played a central role in human culture, and the development of new technologies has significantly impacted how we create, consume, and share music. Bringing music to the computer has been advancing for over half a century. Multiple scientific branches, including musicology, mathematics, computer science, signal processing, and psychology, have been united in numerous ways, creating an impactful collision of art, science, and technology. This interdisciplinary growth has constantly been moving and challenging. The introduction of AI and ML has proved to be a significant milestone.

There are several approaches to AI music generation, ranging from rule-based systems that follow predetermined musical structures to machine learning algorithms that learn patterns in existing music and use this knowledge to generate new compositions [3]. Machine learning algorithms, in particular, have shown great promise in the generation of music, as they can analyze large amounts

of data and generate new compositions that are coherent and follow the conventions of a particular genre or style.

Algorithmic composing using formalizable methods—Guido of Arezzo was the first to contain the music theory, mainly known as “solfeccio”, in 1025 [4]. He created a method to automate the conversion of text into melodic phrases. Electronic instruments communicate with computers through a Musical Instrument Digital Interface (midi) protocol containing information such as pitch and duration. We can use midi for the composer to compose an orchestra without having the orchestra itself. We could create every part of the orchestra from midi with a synthesizer.

MIDI is a protocol for playing, editing, mixing, and recording music. MIDI note notation maps note names to a certain number. It makes it easy to represent music and analyze music using a computer. The basic form of music is composed of different notes in various ways to form a pleasing sound. So the musical synthesizer also used a midi file while composing, mixing, and recording song audio.

In the early 1950s, Iannis Xenakis used the concepts of Statistics and Probability to compose music – popularly known as Stochastic Music [5]. He defined music as a sequence of elements (or sounds) that occurs by chance. Hence, he formulated it using stochastic theory. His random selection of elements was strictly dependent on mathematical concepts. Instead of using specific mathematical concepts, we use a model to train and generate on a probabilistic basis concerning previous notes and trained models.

An audio file has three main parts explained below:

- **Pitch:** Pitch explains the measure of how high or low a sound is.
- **Notes:** A, B, C, D, E, F, and G are the notes included in the music.
- **Octave:** The octave explains the pitch range of any note.

We can change the music can by transposition. Transposition is changing the pitches of a musical note by certain intervals. In transposition, the key changes, but the musical content remains the same. We can generate new music by creating a sequence of notes that is pleasing to hear. This project also focuses on generating music based on previous notes.

This project focuses on generating music automatically using Recurrent Neural Network (RNN) and LSTM. We do not necessarily have to be music experts to generate music; even a non-expert can generate decent-quality music using RNN. Being able to generate decent music using AI is a giant leap for the industry of music.

1.2 Problem Statement

Several techniques have been proposed and implemented to enable automatic music generation using machine learning, such as recurrent neural networks (RNNs) [6] and generative adversarial networks (GANs) [7]. While some systems have been developed to generate music based on user input, the results are often unpredictable. The generated music lacks the complexity and originality of human-generated music. There is still a need to improve the accuracy and variability of the generated music.

One challenge is the need to capture music's complex and diverse structure, including melody, harmony, rhythm, and timbre, and generate creative and original music rather than simply reproducing existing compositions. The project will involve researching, designing, and testing machine learning algorithms and techniques with algorithmic approach of generating music. It involves the novel concept of using AI music with algorithmic generated music to create pleasing music.

1.3 Objectives

The objectives of this project are as follows:

- Understand musical theory and symbolic music representation.
- Analyze music by pre-processing it into a symbolic representation and handle time-series data and prediction using Neural Networks.
- Train a model on a large dataset of musical compositions to learn genre or style-specific features and patterns.
- Implement an interface for users to input their desired instrument and genre to receive a generated musical composition as output.
- Evaluate different directions music can go using prediction and model network.
- Use the concept of the tone-matrix algorithm to generate music based on given image intensity.
- Developing a novel concept that combines AI music with algorithmic-generated music to create pleasing and unique compositions.

1.4 Scope

This project focuses on generating music using computers without the involvement of musicians. As technology advances from simple calculations to diagnosing patients, we can also add its scope to the musical industry. Instead of going to a studio to develop music, we can use AI to generate music for our songs.

Music, as one of the core inheritances since ancient times, has the effect of healing mental trauma, expressing one's feelings, providing mood to work, and many more. Music is more than just an art form; it is a richly rooted culture with amplified importance in today's technologically driven and socially-diverse world. Music can be one of the finest ways to connect people. So automatic music generation is a project that can bring computer AI closer to human beings.

The gaming industry and content creation space require diverse musical sounds to engage with their audiences. However, copyright issues can make it challenging to access and utilize copyrighted music. Automatic music generation projects can help solve this problem by allowing for the creation of diverse, copyright-free music.

Incorporating automatic music generation technology in the gaming industry and content creation space can enhance customer engagement and retention. By generating music that is unique and diverse, businesses can create an immersive and engaging experience for their audience. This can ultimately lead to a more successful and profitable business model.

Automatic Music generation can improve the music industry by producing different musical sound effects in melody and bringing various ways to enhance current musical culture. Automatic music generation helps newbies with no expertise and music experts as they can get new ideas from generated random music. It can also help people generate music using an algorithmic approach and compare its output with the generated output from the Neural Network model. It can also work in music arrangement by using algorithms to rearrange existing pieces of music to create new versions or remixes. This is done by changing the original piece's instrumentation, tempo, or structure or adding new elements, such as vocals or effects.

As musical instruments are expensive to buy and need a team of musicians to create music, we can use AI to generate music for songs. It makes creating music fast and cost-effective within a limited time. Variability, interactivity, and creativity are the major challenges and remarkable potentialities in the functioning of such a model.

2. Literature Review

While we speculate music generation is as old as human society existing for utilitarian to celebratory purposes, it's been a short while since the wave of automatization has influenced this faculty. For a few decades, various research and projects have been carried out for this sole purpose. It is making music-related actions from classification and completion to generation more computerized. And, surely, this endeavor will be a long, complex one with results or, even more importantly, great utility and influence. Even the imagination of an intelligent, hugely automated (if not wholly) dynamic music system is inspirational.

We can see many music-automatizing types of research, projects, and apps. We can divide the deep learning-based computer systems that generate music into two basic categories [8]:

- Autonomous music-making systems (like Amper Music and Jukedeck) that produce creative music for commercials and documentaries.
- Computer-based settings that support live musicians (e.g., FlowComposer and OpenMusic)

Like the more conventional and hugely digitized, automated study subjects such as computer vision and NLP, music is sequential, multiple-channeled, and explorable with other variables. But it has proven to be challenging to work on music due to the following reasons [9]:

- It is temporally hierarchical and dependent.
- Interdependence of an innumerable number of instruments.
- Multiple and complexly interacting basic variables. Jooya Teaching Resources enumerates eight elements of music: timbre, form, texture, rhythm, melody, harmony, tonality, and dynamics.

Human intelligence and creativity are at their peak during music inception, generation, or comprehension. It is a very complex development: artificial structuralization of music in its totality rather than the quality of quantitative information acquisition, analysis, and control. It is a subtle phenomenon, an abstractly scalable growth. It is understood that musical inspiration in humans,

even in simplest forms, is un-exactly-generalizable. That doesn't influence making a system more and more effective as an intelligent tool for musical susception and creation. There has been much research on the audio domain. From text-to-speech to AI assistants, we can see many changes brought about by developing AI. The development in this field also signifies we are close to making an intelligent machine. Music generation also falls under the audio domain since we mainly deal with musical sound generation. Many approaches are proposed for the generation of music and similar projects.

From the Dice Game by Mozart in 1787, the use of musical grammar (library with comprehension of necessary knowledge for proper combinations of musical elements), use of concepts of Probability and Statistics in the early 1950s (stochastic theory) for music generation, we've come a long way today with currently present architectures of deep learning. In the 1970s, the search for musical grammar was probably the most important development in computer-assisted music analysis. While computer-assisted analytical approaches drawing on statistics and information theory developed further, many research projects conducted during the 1980s aimed at the development of new methods of computer-assisted music analysis. Computer-assisted analytical methods shifted from statistical methods to methods drawn from Artificial Intelligence and cognitive sciences in the 1990s.[10] Earlier music generation techniques have used RNNs incorporating temporal dependency. Skuli(2017) and Nelson(2020) used LSTMs to generate music. Convolutional neural networks (CNNs) have recently been successfully utilized to create music, with DeepMind demonstrating the efficiency of WaveNet[11] in 2016, which employs dilated convolutions to produce raw audio. MidiNet was developed by Yang (2017) and employed Deep Convolutional Generative Adversarial Networks (DCGANs) to produce multi-instrument music sequences that can be conditioned on both the chord of the current bar and the music from the preceding bar. With MuseGAN, which uses several generators to produce synthetic multi-instrument music that respects inter-dependencies between instruments, Dong advanced the GAN concept in 2017. Dong employed the Wasserstein-GAN with Gradient Penalty (WGAN-GP) for more regular training.[12] Lastly, attempts have been made to use transformers for music production, like how the most recent developments in NLP have been done with attention networks and transformers. Shaw (2019) developed MusicAutobot, a multi-task engine that can produce new music and construct harmony dependent on other instruments. It combines BERT, Transformer-XL, and Seq2Seq [13].

Following is a brief overview of major works in musical automatization:

2.1 Magenta

Magenta is Google Brain Team's open-source deep learning research project working to explore creativity in sketches, images, and music. It was originally envisioned to explore other sides of developing algorithms that can work as a tool to generate compelling art potentially. It has built a wide community of coordinating artists, coders, and ML researchers. The core team built open-source infrastructure around Tensorflow to make art and music, the field and subject exploratory, exciting, and useful.

It was open-sourced in June 2016 and implements a regular RNN and two LSTMs. It can handle any monophonic midi file, and the documentation is good, so it's easy to set up.

Magenta has provided the model trained on tens of thousands of midi files. The advantage of employing these pre-trained models is that we can start producing new midi files immediately. Magenta can now only produce a single stream of notes, although efforts have been made to add piano, guitar, and drums to the created melodies.

The first blog on Magenta [14] shared the following outline of where magenta was headed to:

- Generation: Algorithmic
- Attention and surprise: Artistic
- Storytelling: Combining generation, attention, and surprise, thereby generating a compelling story
- Evaluation: With evolution, from a suitable audience.
- Other Google efforts: The Artists and Machine Intelligence (AMI) project to connect and communicate how the two disciplines can interrelate.

Magenta first released a music-generating RNN in TensorFlow after converting the MIDI files to a format understandable to TensorFlow. This release predicted the next note to a given sequence of notes by learning probability distribution theorems and analyzing patterns, creating even an entire melody (unsupervised). Derivation of labels and trying to predict the next note in a sequence made the model supervised.

This was followed by using lookback RNN or attention RNN (LSTM) to generate long-term structure in songs, closely followed by connecting a TensorFlow model to a MIDI controller and synthesizer in real-time. Gradually, the use of Reinforcement Learning(RL) to tune RNN, A.I.Duet

(an interactive experiment allowing duet with a computer by Google Creative Lab, NSynth (Neural Audio Synthesis), JavaScript API called Magneta.js to help generate music and art on the web, interactive exploration of music styles through Multitrack MusicVAE, and many more with gradual and remarkable progress in terms of quality of product, the intersection of creativity and machine. The latest reports of magenta talk outputs like DDSP-VST (neural audio synthesizer for digital audio workstation) and auto-regressive long-context music generation with Perceiver AR(generates music with long-term coherence and structure in both symbolic and audio domains). Magenta has proved fairly effective in its mission, evolving further as a remarkable pioneer in automatic music generation [15].

2.2 Deep Jazz

Built-in a thirty-six-hour hackathon by Ji-Sung Kim, deep jazz uses two libraries for deep learning: Keras and Theano, to generate jazz music. It uses a two-layer LSTM that learns from a midi file as its input [16]. It is thus a computational music project to create original jazz compositions using AI. DeepJazz introduces several techniques to enhance the quality of generated music, including a melody harmonization strategy, a chord progression mapping technique, and a note recombination mechanism.

2.3 Aiva

Artificial Intelligence Virtual Artist (AIVA), founded in February 2016, focuses on writing classical and symphonic music, generally emotional soundtracks. It became the first virtual composer in the history of the world to be honored by a music society (SACEM). AIVA can recognize musical patterns by reading a sizable collection of previous classical music compositions (authored by human composers like Bach, Beethoven, and Mozart) and, on this basis, writing on its own. Deep learning and reinforcement learning architecture form the foundation of the AIVA algorithm. Since January 2019, the business has made its commercial product, Music Engine, available. It can create short compositions in various styles that last up to three minutes (rock, pop, jazz, fantasy, shanty, tango, etc.).AIVA could grasp the principles of music theory and comprehend the art of music composition thanks to learning from the great figures in musical history.

Additionally, it aided AIVA in building a mathematical model representing what music is. Aiva then employs this model to create wholly original music. AIVA utilizes CUDA, TITAN X Pascal GPUs, cuDNN, and the reinforcement of deep learning algorithm principles available in TensorFlow. However, AIVA can only currently write music for the piano; orchestration, composition,

and production of the piece all require human expertise. It's also crucial to note that AIVA leverages GPU computing, which has produced a plagiarism checker that can determine whether a track was composed entirely or partially using data from the database AIVA learned from. In addition, additional Turing tests with music professionals were done. Participants concur that it is impossible to differentiate the compositions of AIVA as AI generation or human fabrication [17].

2.4 OpenAI Jukebox

Jukebox is a neural net for music generation as raw audio in various genres and styles. It overcomes the limitations of symbolic music generators like piano roll (specifying the timing, pitch, velocity, and instrument of each playable note), like the inability to capture the human voice and other subtler timbres, dynamics, and expressivity without which music is bland. Direct generation as raw audio is challenging (Very long sequence: At CD quality, a song of 4 minutes has more than 10 million timesteps). Jukebox uses an auto-encoder that compresses music as discrete codes to address this long-range dependency problem.

An approach called VQ-VAE based on quantization is used in which audio is compressed to a discrete space. Architecture is modified to alleviate codebook collapse (use of random restarts), maximize the use of upper levels (use of separate decoders, level-wise independent reconstruction of input), and allow reconstruction of higher frequencies with ease (add a spectral loss to penalize norm of the difference between input and reconstructed spectrograms). Three levels of VQ-VAE are used to compress audio by 8x(bottommost level, which produces the best reconstruction), 32x, and 128x (top level, which retains only essential information), respectively, wherein much of the audio detail is lost, it sounds noisier but essential information on pitch, timbre, and volume are retained.

Further, transformers are used to generate codes. Prior models are taught the music code output by VQ-VAE and generate music in this discrete space. From top to bottom level prior with decreasing value of compression. Outputs of top-level prior used for long-range music structure have lower audio quality than others but can capture high-level semantics. These are trained using a simplified variant of Sparse Transformers as autoregressive models, each having 72 layers of factorized self-attention in a context of 8192 seconds, corresponding to 24, 6, and 1.5 seconds of raw input at the top-to-bottom levels, respectively.

Codes are generated using priors, unsampled by un-samplers, and decoded back into raw audio space using the VQ-VAE decoder. Songs are thus sampled. They curated a new dataset of 1.2 million songs from the web paired with corresponding lyrics and metadata (artist, album, genre,

year of songs, common moods, or playlist keywords associated) from LyricWiki to train the model. They did the training on 32-bit, 44.1 kHz raw audio, and random downmixing of right and left channels was done as data augmentation to produce mono audio. The model clusters the similarity of data and metadata after learning in an unsupervised way, thus enabling the functionality of artist and genre conditioning. Furthermore, the encoder-decoder attention layer is structured to align lyric music, thus enabling lyric conditioning.

From remarkable gap with human-created music, presence of discernable noise, slow (not usable for interaction), absence of inclusivity of music types in the dataset, and more, there are limits to what the Jukebox can do; the team is continuously efforting to expand the musicality, other features, connectivity, inclusivity in this exciting creative space [18].

2.5 MusicBERT

MusicBERT is a large-scale pre-trained model for Symbolic music understanding developed by Microsoft Research Asia[19]. It focused on understanding music through its symbolic representation(such as MIDI format). MusicBERT was inspired from BERT (Bidirectional Encoder Representations from Transformers) architecture, which has been used in natural language processing tasks. Due to the presence of additional structural elements in music (such as bars and positions) and a wider range of diverse information (such as tempo, instruments, and pitch), using pre-training techniques from natural language processing (NLP) on symbolic music data may result in only minimal improvements in performance. To improve pre-training techniques for symbolic music data, the MusicBERT paper proposes several mechanisms, including OctupleMIDI encoding and bar-level masking strategy [19]. MusicBERT has various applications in music, such as melody completion, accompaniment suggestion, genre classification, and style classification.

MusicBERT has developed a novel encoding method, OctupleMIDI method [19]. OctupleMIDI encodes notes into tuples with eight elements, representing different aspects of music, including instrument, tempo, time signature, bar position, pitch, duration, and velocity. This reduces the length of the music sequence compared to REMI encoding, which eases modeling music sequences by Transformer. MusicBERT uses a bar-level masking strategy as opposed to the masking strategy in the original BERT for NLP to prevent information leakage while training the model. MusicBERT achieved state of the art performance on the symbolic music understanding task.

2.6 More Researches

Multidisciplinary research and project outputs are observable for music generation using AI taking different input forms. Many physical to cognitive dimensions are found to have abundantly been experimented upon. To state a few:

- Researches on sound generation based on image color spectrum using the recurrent neural network; implementation of transition between color and music characteristics with rationale for choosing have been done. The neural network has generated musical compositions by analyzing the correlation between color and musical characteristics. It begins by determining the most suitable type of neural network for this task, which was found to be recurrent neural networks with long short-term memory (RNN LSTM). It concludes with an experiment in which the model is trained on Beethoven's compositions, and compositions are generated and evaluated by experts for harmony and melodiousness. The results indicate that the program generates melodic compositions but that the model was trained on a small number of compositions [20].
- Based on the Pix2Pix architecture, a conditional GAN sound is generated from an image by successively labeling and pairing images and sounds from various points in time and styles, extracting shared features from the data by investigating several techniques for music feature extraction, adding multi-modal layers to the GAN and use of a technique for composing original musical compositions implementing the features generated for generation of sound. Thus the composition of music this way exemplifies a solution to the question: How does that image sound? [21]
- A framework to generate music corresponding to the emotion of the person predicted by a model divisible as the Image Classification Model (identification and classification of the facial expression into one of the seven major sentiment categories: Happy, Disgust, Angry, Fear, Sad, Surprise and Neutral by using CNN) and the Music Generation Model (essentially a Doubly Stacked LSTM architecture to generate corresponding music) [22].
- Generation of music from images connecting the visual and auditory domains using emotion. The resulting music enriches visual art and provides a form of translation to facilitate the perception of visual art without relying on the visual system. They used pre-trained image representations, RNN, and Transformer architectures to build underlying models [23].
- MelNet is a generative model for audio in the frequency domain that DeepAI introduced in 2019. It was based on a variant of Transformer architecture that introduces the idea of

recurrence to the deep self-attention network [24]. MelNet was trained on a large raw audio dataset, allowing it to generate a wide range of sounds and styles. Additionally, its multi-scale architecture enables it to generate audio with a high temporal resolution, meaning that it can generate audio that is more realistic and less choppy than other models

MelNet creates high-resolution spectrograms with realistic structures on both local and global scales by combining a highly expressive autoregressive model with a multiscale modeling approach. MelNet is considered one of the state-of-the-art models for speech synthesis (music generation is also its part), with its ability to generate high-quality audio, diverse styles, and genres and its application for other use cases such as speech and environmental sounds.

- In 2016, researchers at Google DeepMind introduced WaveNet[11], a deep neural network architecture for generating raw audio waveforms. The WaveNet architecture is based on a fully convolutional neural network (CNN) that uses dilated convolutions to increase the receptive field of the network without increasing the number of parameters. The model is fully probabilistic and autoregressive, meaning that the predictive distribution for each audio sample is conditioned on all previous ones. When applied to text-to-speech, WaveNet achieves state-of-the-art performance[11].

When used to model music, WaveNet generates novel and often highly realistic musical fragments. For the production of musical sound, it was found that enlarging the receptive field was crucial. But, Even with a receptive field of several seconds, the models did not enforce long-range consistency, resulting in second-to-second variations in genre, instrumentation, volume, and sound quality. Nevertheless, the samples were often harmonic and aesthetically pleasing, even when produced by unconditional models. WaveNet[11] can also be used as a conditional music model, generating music given tags specifying, for example, genre or instruments. The paper reported using the MagnaTagATune dataset, which was cleaned up by merging similar tags and removing those with too few associated clips[11]. The results were reasonably good.

- The research paper titled "DeepJ: Style-Specific Music Generation" presented an end-to-end generative model that can compose music conditioned on a specific blend of composer styles. The paper employed the Biaxial LSTM architecture to generate musically plausible results with measure-level structure[25]. The dataset utilized in this study consisted of music from various composers, which defined a specific artistic style. The paper advanced the Biaxial model by incorporating volume and style, resulting in improved quality of the generated music.
- In the paper "MIDINET: A CONVOLUTIONAL GENERATIVE ADVERSARIAL NET-

WORK FOR SYMBOLIC-DOMAIN MUSIC GENERATION”[26], the authors introduce a model for generating MIDI files using a combination of convolutional neural networks (CNN) and generative adversarial networks (GAN). The architecture used in this paper is similar to that of WaveNet. Unlike WaveNet[11], which work on raw audio files, this paper focuses on understanding music from a symbolic representation of music (ie, MIDI file). The proposed model includes a conditional mechanism that leverages available prior knowledge, enabling the generation of melodies from scratch, by following a chord sequence, or by conditioning on the melody of previous bars. This approach allows for greater control and flexibility in generating music.

- The research paper, ”Automatic Generation of Music with Recurrent Neural Networks” by François Pachet (2013), proposes a method of generating music using recurrent neural networks (RNNs). The author trains an RNN on a dataset of MIDI files and generates new music by sampling from the learned probability distribution.

The paper begins by discussing the importance of melody and harmony in music and how these elements can be generated using machine learning techniques. The author then introduces RNNs as a suitable model for music generation due to their ability to handle sequential data.

The training data for the RNN is a dataset of MIDI files, which contain information about the pitch, duration, and velocity of notes played in a music piece. The RNN is trained to predict the next note in a sequence given the previous notes. The model is trained using a maximum likelihood estimation approach and the cross-entropy loss function.

To generate new music, the RNN is fed with a sequence of notes, and it samples from the learned probability distribution to predict the next note in the sequence. The author uses a temperature parameter to control the randomness of the generated music. Higher temperatures produce more random music, while lower temperatures produce more predictable music.

The paper evaluates the model by conducting a subjective test where human listeners are asked to rate the quality of the generated music. The results show that the generated music is perceived as coherent and interesting by human listeners.

One of the advantages of the proposed method is that it generates music with a natural-sounding structure and melody. However, the paper also notes some limitations of the method, such as the inability to generate long-term structure and the tendency to produce repetitive patterns.

So, the paper demonstrates the effectiveness of RNNs in generating music and highlights

some of the challenges that remain to be addressed in automatic music generation. The method proposed in the paper can be extended and improved using more advanced neural network architectures and training techniques[27].

- The paper "Classical Music Generation with Neural Networks" by Gaëtan Hadjeres and François Pachet (2016)[28] proposes a method of generating music using a combination of recurrent neural networks (RNNs) and deep belief networks (DBNs).

The authors use a dataset of MIDI files and train an RNN to predict the next note in a sequence. They then use a DBN to model the distribution of notes in the training set, which allows them to generate new music that is coherent and interesting. The DBN captures the long-term dependencies and patterns in the music, while the RNN handles the short-term dependencies and generates new sequences[28].

The paper evaluates the model based on a subjective test where human listeners rate the generated music based on its coherence and interest. The results show that the generated music is perceived as more interesting and coherent than randomly generated music.

One of the strengths of this approach is that it can capture the complex structure of music and generate music that is not only coherent but also interesting. The DBN allows the model to capture the high-level structure of the music, such as chord progressions and melodic motifs, while the RNN generates variations and fills in the details.

Another advantage of this approach is that it can generate music in multiple styles. The authors show that the model can generate music in different styles, such as classical, jazz, and pop, by training the model on different datasets.

One limitation of this approach is that it requires a large dataset of MIDI files to train the model. The authors use a dataset of over 23,000 MIDI files, which may not be feasible for some applications. Additionally, the model is trained offline, and generating music in real time may require significant computational resources.

The paper by Hadjeres and Pachet presents a promising approach to music generation using a combination of RNNs and DBNs. The model can generate music that is both coherent and interesting and can be trained on different datasets to generate music in multiple styles. However, it requires a large dataset and may not be suitable for real-time music generation[28].

3. Related Theory

3.1 MIDI

MIDI (Musical Instrument Digital Interface) is a standard protocol for communicating musical information between digital devices, such as computers and synthesizers[29]. It allows different devices to connect and exchange information about musical events, such as pitches, rhythms, and timbres. MIDI data consists of messages that specify various types of musical information, such as the pitches and duration of notes, the tempo of a piece, and the values of various control parameters (e.g., volume, panning, pitch bend).

Each MIDI file is a collection of tracks containing events. Events represent notes on/off, tempo, instrument type, meta-information about the music, etc. Also, each event contains a tick number and a payload. The tick number is the lowest level of time resolution. The tick number of each event describes how many ticks passed from the previous event to the current one. The notion of time in music is defined using the tick, tempo, and resolution. Tempo is the number of beats per minute (BPM), also called Quarter note per minute (QPM). The resolution is the pulses per quarter note (PPQ), thus describing how many ticks per quarter note are being played past. The higher the resolution, the shorter in time one tick.

3.2 Music Theory

Music theory is the study of the principles and practices of music. Music theory includes the study of scales, modes, keys, notes, and chords, as well as the rules for combining these elements to create musical compositions. It also includes the study of musical notation, the system used to represent music in written form, and the study of musical instruments and their characteristics.

3.2.1 Notes

A note is a unit of symbolic representation of a musical sound. This representation is specified by the following primary features, and there are various ways to express the value of each feature:
[30]

- Pitch
 - How high or low a note sounds depends on its pitch. In Hertz (Hz), the note's frequency determines its pitch.
 - Pitch is specified by the position vertically on a score.
 - Pitch notation consists of a number (often notated in subscript) denoting the pitch class octave that is a part of the [-1,9] discrete interval, together with the name of the musical note, e.g., A, A#, B, etc., which is its pitch class.
- Duration
 - Absolute values give it, measured in milliseconds(ms) or
 - It is defined as a relative value, expressed as a fraction or multiple of a reference note's duration, i.e., the whole note \circ . Some examples are a quarter note \bullet and half note \circ .
- Dynamics
 - It is specified by absolute and quantitative value, given by decibels (dB).
 - It is also specified by qualitative value, a notation on a score describing how to play a note that is part of the discrete set [*pp*, *mp*, *p*, *mf*, *f*, *ff*], ranging from pianissimo(softest) to fortissimo(loudest).

3.2.2 Chords

A simultaneous-sounding harmonic set of three or more notes forms a chord. Chords are classified based on the notes and the extent to which they define the chord's quality or by inversion, i.e., the order of stacking notes. Triads, chords with three distinct notes, are the most frequently encountered chords. From a root note, chords are constructed (also known as the starting note). A chord's root note, typically the note with the lowest pitch, serves as its foundation. The type of chord being played will determine the remaining notes. In music notation, chords are typically represented by a symbol called a chord symbol, consisting of the chord's root note and a chord quality. For example, a C major chord would be represented by the chord symbol "C," while a D minor chord would be represented by the chord symbol "Dm."

Chords can also be represented in music notation by showing the notes that make up the chord. This is typically done by writing the chord's notes in stacked thirds, with the root note at the bottom. For example, a C major chord would be represented by the notes C, E, and G, stacked on top of each other, with the C at the bottom.

3.2.3 Time Signature

The time signature determines the rhythmic feel of a piece of music. The time signature is usually written at the beginning of a piece of music and applies to the entire piece unless it is changed later. It is important to follow the time signature when reading and interpreting music notation, as it determines the duration and placement of the notes in the music.

A time signature is typically expressed as a fraction, with two numbers written vertically. The top number of the fraction indicates the number of beats in each measure. The bottom number represents the type of note that receives one beat. For example, in the time signature 4/4, the numerator indicates that there are four beats in each measure, and the denominator indicates that a quarter note receives one beat.

3.2.4 BPM

Beats per minute (BPM) measures the tempo of a piece of music. The BPM value indicates the number of beats that occur in one minute of music. Tempo indicates the pace of the musical composition. Music with a fast tempo, such as dance music, tends to be energetic and lively, while music with a slow tempo, like ballads, tends to be more mellow and relaxed. It is also important to consider when mixing and mastering music, as it can affect how it sounds when played back at different speeds.

3.2.5 Sheet Music

Sheet music is a written representation of music. It is a series of symbols and notations that indicate how a piece of music should be played. Musicians use sheet music to learn and perform music, which is an important tool in creating and disseminating music. Sheet music is typically written on a staff, with five horizontal lines and four spaces on which musical notation is written. Each line and space on the staff corresponds to a specific pitch, with higher pitches written higher and lower pitches written lower on the staff. Sheet music also includes symbols and notation that indicate the duration of notes (how long they are held for), the rhythm of the piece, and other musical elements such as dynamics (loudness and softness) and articulation (how the notes are played). Fig (3.1) shows a diagram of notes, rests, names, and values in music theory.












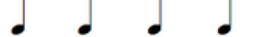



Name	Note	Rest	Beats	$1 \frac{4}{4}$ measure
Whole			4	
Half			2	
Quarter			1	
Eighth			$\frac{1}{2}$	
Sixteenth			$\frac{1}{4}$	

Figure 3.1: Notes and rests symbol[2]

3.3 Music21

Music21 is introduced as "a toolkit for computer-aided musical analysis and computational musicology". It is a Python-based project containing a collection of specialized tools and objects for music-related general-purpose functions like the study of music, generation of music, analysis of the symbolic representation of music, and more. It is used for manipulating and analyzing data one already has or is generating. It doesn't have a native data format and can work with many common formats, even extendable to work with more. Based on the type of data, storage of music21 data, which is not generally required, can be done in various formats corresponding to the type of data storable.

Music21 works by manipulating symbolic musical data, which is further used to synthesize musical sound waves. Thus, sound wave generation, exploration, and studies are facilitated by music21.

3.4 Tone Matrix

The tone matrix is a 16×16 sequencing grid consisting of 256 cells. Each row in the grid represents different notes, and each column represents time. In simple term, the vertical axis represents frequency in increasing order and the horizontal axis represent the time axis. A cell in a tone matrix gives us frequency at a given time. It allows to create the sound patterns by turning on and off cells in a grid. It can be used to create short musical patterns or loops that can be repeated and combined

to create longer pieces of music. The 16x16 grid size is a common format in music sequencing software and hardware, as it provides a large enough space to create complex patterns while remaining manageable and intuitive to use.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C6																
A5																
G5					x							x				x
F5					x			x	x							
D5				x		x		x	x	x						
C5		x					x					x			x	
A4										x			x			x
G4				x												
F4				x						x						
D4																
C4			x								x			x		
A3	x															
G3		x							x						x	x
F3			x										x			
D3																
C3	x															

Figure 3.2: Example of Tone Matrix with ON (x) and OFF cells

Images of any dimension are provided as input. It's converted to a black-and-white image of specific pixels. Then, the image is mapped with the parameters of the tone matrix. The cells with white pixels are referred to as the ON cells. Finally, the music is obtained.

Our approach to creating the tone matrix from an image has potential applications in music, art, and multimedia fields. It offers a new and creative way for individuals to express themselves through sound and allows for a more dynamic and immersive experience for the audience.

Using an image to create a musical sequence can be an innovative and creative approach to music composition and a useful tool for musicians and producers. Using visual patterns to generate musical sequences, new and unique compositions can be created that might not have been possible using traditional methods.

3.5 RNN and LSTM

A Recurrent Neural Network can be defined as the generalization of a feed-forward neural network having an internal memory for performing the same function for every input data. In contrast, the output of the current input depends on the past computation. In other neural networks, all inputs are independent, but RNNs can use memory's internal state to process related sequences. The basic block diagram of RNN is shown in the figure below.

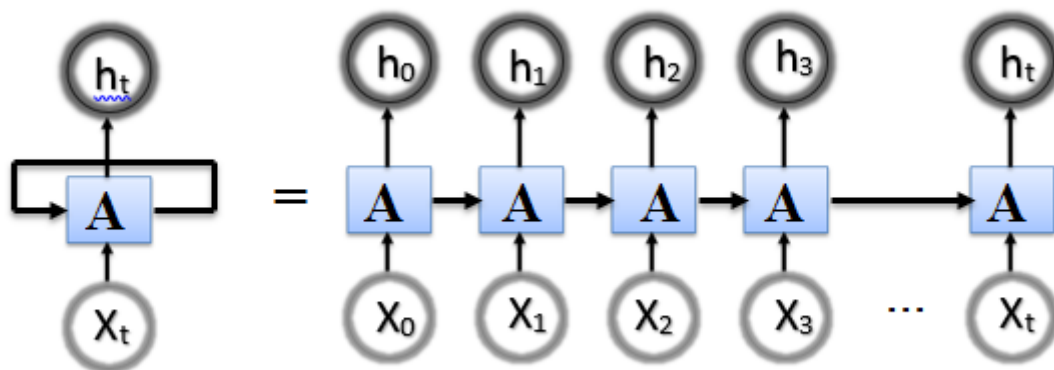


Figure 3.3: RNN

As RNNs cannot memorize data for a long time, Long Short Term Memory (LSTM) solves the long-term dependency problems of RNNs. The Vanishing Gradient problem can be solved using LSTM. The model minimizes the loss after every time step by calculating loss concerning some weights to get the best result. LSTM deals with vanishing and exploding gradient problems by introducing new gates, such as input and forget gates, which allow for better control over the gradient flow and better preservation of 'long-range dependencies'. [31]

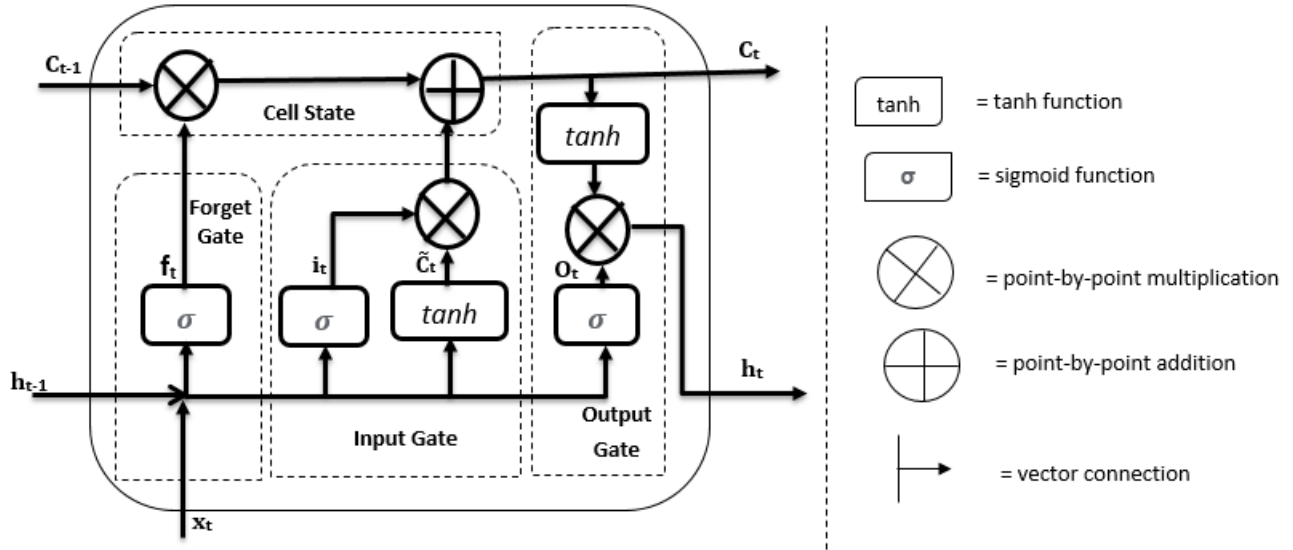


Figure 3.4: LSTM

LSTM solves the problem of long-term dependency. LSTM uses a forget gate to replace data stored in memory cells if it is not needed in further prediction. Unlike RNN, LSTM consumes more memory as it stores data for a long period, but it is more suitable for solving problems that require data dependency in prediction. This project needs LSTM mainly because the generation of music generally depends on the dependency of earlier notes more accurately to generate more artistic music.

3.5.1 Forget Gate

The forget gate f_t is the first block in the LSTM design. The sigmoid activation function receives data from the current input X_t and the previous hidden state h_t . Past data will be forgotten if the output value is close to 0 and maintained but hidden if the output value is close to 1. This gate is employed to remove unimportant data from the model training process. The operation performed in forget gate is given by[31]:

$$f_t = \sigma (W_f \cdot [h_t - 1, x_t] + b_f) \quad (3.1)$$

where,

t = time-step

f_t = forget gate at t

x_t = input at t

h_{t-1} = Previous hidden state

W_f = Weight matrix between forget gate and input gate

b_t = bias at t

3.5.2 Input Gate

The input gate is responsible for the quantification of the importance of new information. In the input gate, the second sigmoid function receives two arguments: the current state X_t and the previously hidden state h_{t-1} , which transforms the value between 0(important) to 1(not important).

The tanh function will receive identical data from the hidden and current states. The tanh operator will build a vector \tilde{C}_t containing every possible value between -1 and 1 to control the network. The output values produced by the activation functions are ready for multiplication point-by-point. Mathematically[31],

$$i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \quad (3.2)$$

$$\tilde{C}_t = \tanh (W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3.3)$$

where,

t = time-step

i_t = input gate at t

W_i = Weight matrix of sigmoid operator between input gate and output gate

b_t = bias vector at t

\tilde{C}_t = value generated by \tanh

W_c = Weight matrix of tanh operator between cell state information and network output

b_c = bias vector at t, w.r.t W_c

3.5.3 Cell State

The input gate and forget gate have provided the network with sufficient information. Making a decision and storing the data from the new state in the cell state comes next. The forget vector f_t multiplies the previous cell state C_{t-1} . Values will be removed from the cell state if the result is 0. The network then executes point-by-point addition on the output value of the input vector i_t , updating the cell state and creating a new cell state C_t . Mathematically[31],

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (3.4)$$

where C_t , f_t , i_t and \tilde{C}_t are cell gate, forget gate, input gate, and value generated by tanh at time t respectively and \tilde{C}_{t-1} is cell state information in time $t - 1$.

3.5.4 Output Gate

The output gate regulates the hidden gate h_t . The current input X_t and prior hidden state X_{t-1} are supplied to the sigmoid function, which is then multiplied with the output of the tanh function to obtain the present hidden state. The final outputs from a traditional LSTM unit are the current state C_t and the current hidden state h_t . Mathematically[31],

$$O_t = \sigma (W_o [h_t - 1, x_t] + b_o) \quad (3.5)$$

$$h_t = O_t * \tanh (C_t) \quad (3.6)$$

where O_t and h_t are output gate and LSTM output at time-step(t). W_o and b_o are the Weight matrix of the output gate and bias vector, w.r.t W_o , respectively.

3.5.5 Activation function

The **sigmoid function**, also known as a squashing function, has a set of all real numbers as its domain, whose range is between 0 and 1. As a result, the output is always between 0 and 1 regardless of whether the function is given a very big positive or very large negative value as input. The sigmoid function is used in LSTM networks as a gate activation function to control the flow of information through the cell state.

In an LSTM cell, the sigmoid function is used in the input gate and forget gate to decide whether to allow new input into the cell state and whether to forget the previous cell state information. The output gate of the LSTM cell also uses the sigmoid function to decide how much of the cell state to output to the next time step.

The sigmoid activation function [32] is given by:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3.7)$$

where x is the input to the function.

The **tanh activation function** role is similar to the sigmoid function. It has a domain of a set of all real numbers, and unlike sigmoid, its range is between -1 to 1. In LSTM neural networks, the hyperbolic tangent (*tanh*) activation function is used in the cell state update equation and output gate. In the cell state update equation, the *tanh* function is used to scale the candidate values and

control the cell state information flow. It squashes the input values between -1 and 1, which helps to prevent the cell state from exploding or vanishing during the training process.

In the output gate, the \tanh function is used to squish the cell state value between -1 and 1 before multiplying it with the output gate value. This ensures that the output value is also bounded and does not grow too large or small. The \tanh function also enables the LSTM to selectively forget or remember information based on the input and previous state values.

The tanh activation function [33] is calculated as follows :

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.8)$$

where x is the input to the activation function.

Softmax is a type of activation function that is commonly used in neural networks, particularly for multi-class classification problems. The softmax function[34] takes a vector of real-valued inputs and returns a probability distribution over the possible output classes.

The function works by exponentiating each input value and then normalizing the resulting values so that they sum to 1. This produces a set of values between 0 and 1 that can be interpreted as the probabilities of each class being the correct output.

The softmax function[34] can be expressed mathematically as:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (3.9)$$

where \mathbf{z} is the input vector of length, K is number of classes in the multi-class classifier, and $\sigma(\mathbf{z})_i$ is the output probability for class i , e^{z_i} is standard exponential function for input vector. The sum in the denominator ensures that the output probabilities sum to 1.

3.6 Optimizer

An optimizer is an algorithm used to adjust the parameters of a model to minimize the loss function. The loss function measures how well the model fits the training data, and the goal of training a machine learning model is to find the set of parameters that minimizes the loss function.

Optimizers are a key component of many machine learning algorithms, including neural networks, and they play a crucial role in improving the performance of a model on a given task. The optimizer used in this project is Adam Optimizer. **Adam** (Adaptive Moment Estimation) is a variant

of stochastic gradient descent that uses moving averages of the parameters to provide running estimates of the second raw moments of the gradients; the term adaptive in the algorithm’s name refers to the fact that the algorithm ”adapts” the learning rates of each parameter based on the historical gradient information.

The Adam update rule can be written as follows:

$$\begin{aligned}
 m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\
 v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \\
 \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\
 \hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \\
 \theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t
 \end{aligned} \tag{3.10}$$

where m_t and v_t are the first and second raw moment estimates at time step t , β_1 and β_2 are hyper-parameters that control the decay rates of the momentum estimates, \hat{m}_t and \hat{v}_t is the bias-corrected versions of the momentum estimates with η step-size, and ϵ is a small constant added to the denominator to prevent division by zero.

3.7 Loss

The loss function measures how well the model can predict the correct output for a given input. Training a deep learning model aims to find the set of model parameters(i.e., weights and bias) that minimize the loss function. The loss function used in the project model is KL divergence.

The Kullback–Leibler divergence denoted $D_{\text{KL}}(P \parallel Q)$, is a type of statistical distance: a measure of how one probability distribution P is different from a second, reference probability distribution Q [35]. It is used in machine learning and information theory to quantify the amount of information lost when approximating one probability distribution with another.

Mathematically, KL divergence is defined as:

$$D_{\text{KL}}(P \parallel Q) = \sum_{i=1}^N p(x_i) \cdot (\log p(x_i) - \log q(x_i)) \tag{3.11}$$

where, p and q are two probability distributions.

KL divergence is used as a loss function in machine learning models that are trained to model a target distribution. KL divergence is used in the generative model, as it measures the difference between a target distribution and the generative model and minimizes the difference.

3.7.1 Regularization

Regularization is a technique that is used to adjust loss function to prevent overfitting and underfitting problem in machine learning models. L1 and L2 regularization are mostly used regularization techniques to overcome overfitting and underfitting model.

L1 regularization, also known as Lasso Regularization works by adding a penalty term to the loss function of a model. This penalty term is equal to the sum of the absolute values of the model's coefficients multiplied by a regularization parameter, λ . The greater the value of λ , the stronger the regularization. Mathematically[36],

$$L_1 = \lambda \sum_{i=0}^N |w_i| \quad (3.12)$$

The effect of L_1 penalty is to shrink the magnitude of the model's coefficients towards zero. The coefficients that are shrunk to zero effectively remove the corresponding features from the model, resulting in a simpler and more interpretable model[36].

L2 regularization, also known as Ridge regularization, adds a penalty to the loss function that is proportional to the squared magnitude of the weights. This penalty encourages the model to spread out the weight values more evenly across all features and can help to reduce the impact of individual features on the overall model[36].

$$L_2 = \lambda \sum_{i=0}^N w_i^2 \quad (3.13)$$

Elastic Net regularization is combination of L1 and L2 regularization. It adds a penalty to the loss function that is a combination of the L1 and L2 penalties. Elastic Net regularization is effective in cases where there are many features that are highly correlated with each other, as it can help to identify groups of features that are jointly important for predicting the target variable. The L1 penalty encourages sparsity in the feature weights, while the L2 penalty encourages smoothness and stability. Then the loss function with Elastic Net regularization is given by[36]:

$$Loss = Error(y, \hat{y}) + r\lambda \sum_{i=0}^N |w_i| + \frac{1-r}{2}\lambda \sum_{i=0}^N w_i^2 \quad (3.14)$$

where,

$Error(y, \hat{y})$ is loss function which can be MSE, K1-divergence, etc

$r\lambda \sum_{i=0}^N |w_i|$ is penalty of L1 Regularization

$\frac{1-r}{2}\lambda \sum_{i=0}^N w_i^2$ is penalty of L2 Regularization

r is a hyperparameter that controls the relative weighting of the L1 and L2 penalties.

4. Methodology

Music creation is a complex and multi-faceted field that involves a wide range of factors, including harmony, melody, rhythm, and other elements. One important aspect of music creation is determining the next note that harmonizes with the previous note, as this can have a significant impact on the overall sound and feel of the music. Our model is also based on a prediction of the next notes based on a sequence of the previous note. We used Long Short Term Memory(LSTM) as a core concept for developing a generative model for music generation.

We employed the concept of a tone matrix to generate music through an algorithmic approach, which was inspired by ToneMatrix by Audiotool. Our approach utilized an image to generate a switch for the tone matrix input. By turning the switch ON or OFF based on the intensity distribution in the image, we were able to produce pleasant music.

Our research methodology employs a novel concept that combines AI and algorithmic models to produce aesthetically pleasing music.

The methodology for working on AI generative model is discussed below.

4.1 Block Diagram of Basic Model

The block diagram shown in Figure (4.1) is a general outline of the music generation process using machine learning. It shows how this project was approached. The various block in the diagram is discussed below.

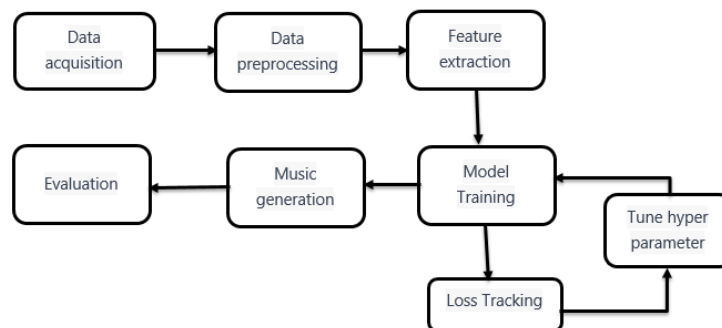


Figure 4.1: Basic Block Diagram of AI Model

4.1.1 Data acquisition

The first step in music generation using machine learning is to collect a dataset of existing music. The dataset can be in various formats, such as MIDI and kern files for solving music generation problems. The dataset used in this project was sourced from the ESAC Folk database[1]. The ESAC (Essen Associative Code) and Folk database is a collection of over 21,000 folk music recordings from various regions across Europe. The database is maintained by the European Broadcasting Union (EBU) and is freely accessible to the public.

A custom dataset of different piano tracks of Hindi music was also used to train in the same architecture of the model while experimenting with the model. The purpose of using this custom dataset was to observe the performance of our generative model when trained with diverse music. The output generated from that trained model was also pleasing to hear.

4.1.2 Data pre-processing

The next step after collecting the dataset is, pre-processing of data. The library used for the analysis and pre-processing of the music files was Music21. Music21 is a Python-based toolkit for computer-aided musicology. Music21 is one of the most used music analysis tools while working with symbolic music representation. The ESAC Folk database[1] contains music data in the .kern format. Humdrum Kern is a music notation file format used for representing musical scores. It is a plain text format that uses a specific syntax to encode musical information, including notes, chords, and other musical symbols.

The music data from the dataset was converted to MIDI format for use in this project. MIDI is a commonly used format for representing music data and is well-suited for use in deep learning models. Midi file is converted to a music21 object using the Music21 library for further analysis and processing.

4.1.3 Feature extraction

After pre-processing of data, the relevant features of the music are extracted from the dataset. Music is composed of different features such as notes, tempo, duration, key, bpm, melody, rhythm, etc. which make out how music is being played. It is important to choose the most important features while training the model. And basic construct of music is notes played and the duration up to which notes are being played.

Since the main feature of the music is its notes, chords, and rest with their duration for occurrence, they are taken as the main feature for the training model. For easier interpretation, the parsed music21 object is first chordify, i.e., notes are changed to chord. Then chords, rest, and duration are extracted as their primary features for further operation.

When the original file is chordify to extract the feature, and that extracted feature is used to reconstruct the original file, there is a slight difference in the generated file since music depends on other factors such as bpm, time-signature, etc. But the difference(loss of information) is not quite significant. The waveform of the raw audio file and its reconstructed file for a duration of 2 seconds is shown below:

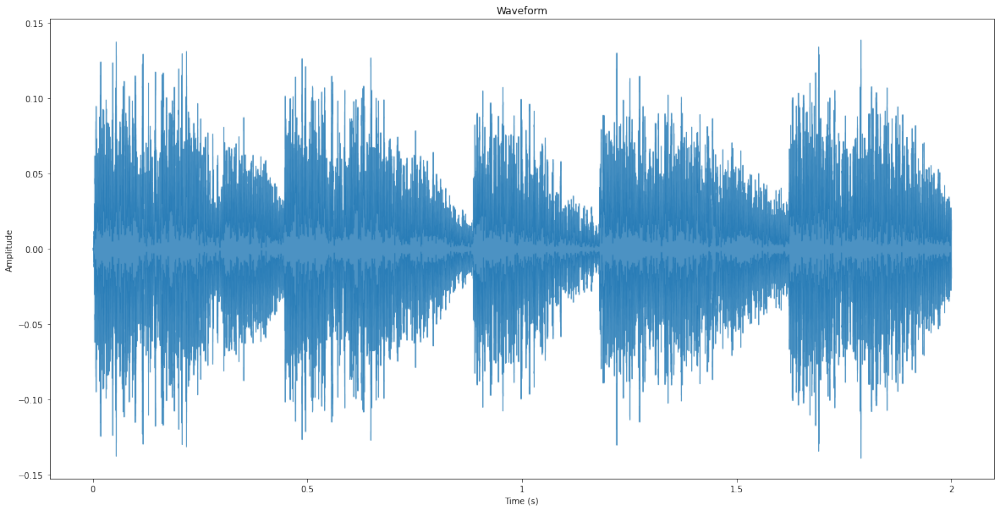


Figure 4.2: Original file waveform of duration 2 sec

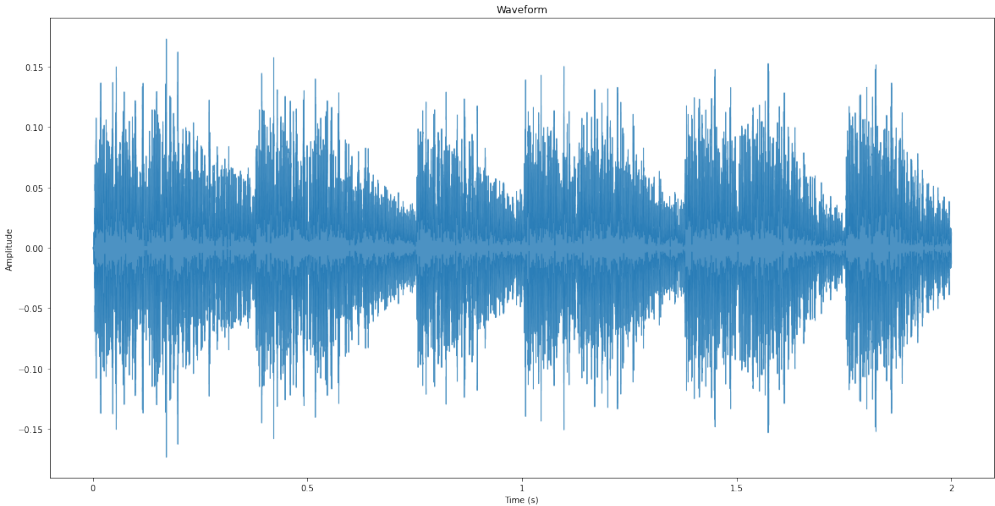


Figure 4.3: Reconstructed file waveform of duration 2 sec

The waveform of a piece of music can be affected by various factors, including the tempo, time signature, and other rhythmic elements of the music. However, these factors may not necessarily have a significant impact on the overall sound of the music when it is listened to by a human.

In this project, the extracted notes and durations are initially encoded by mapping them to a unique integer value, which is then used for training the model. This approach enables the model to process and manipulate the music as a sequence of numerical values instead of symbolic data. The encoding technique employed in this project is one-hot encoding, which involves representing each unique value in the data as a binary vector with a length equal to the total number of unique values. This enables the model to better differentiate between different categories of data.

4.1.4 Model training

Once the data has been pre-processed and the relevant features have been extracted, notes and duration are fed into the embedding layer. The embedding layer represents the notes and duration of the music as numerical vectors that can be input into a model. An embedding layer is a neural network layer that learns a low-dimensional representation of the input data. The embedding layer can learn a representation of the different notes and durations in the music, allowing the model to process and manipulate these elements more meaningfully.

After the notes and duration had been embedded, they were concatenated into a single layer and passed through an LSTM (Long Short-Term Memory) layer. LSTM layers are a type of recurrent neural network layer that is designed to capture long-term dependencies in sequential data. LSTM was used to model the temporal structure of the music and make predictions about the next notes and their duration based on the patterns learned from the data.

For our model, we utilized the Adam optimizer with a learning rate of 0.001 and employed the Kullback-Leibler Divergence (KL-divergence) as the loss function. The model consists of two dense layers - one for predicting notes and the other for predicting their duration in the music. To account for the different outputs, we defined separate loss functions for each, with both utilizing KL-divergence. The operation of the entire model is illustrated below:

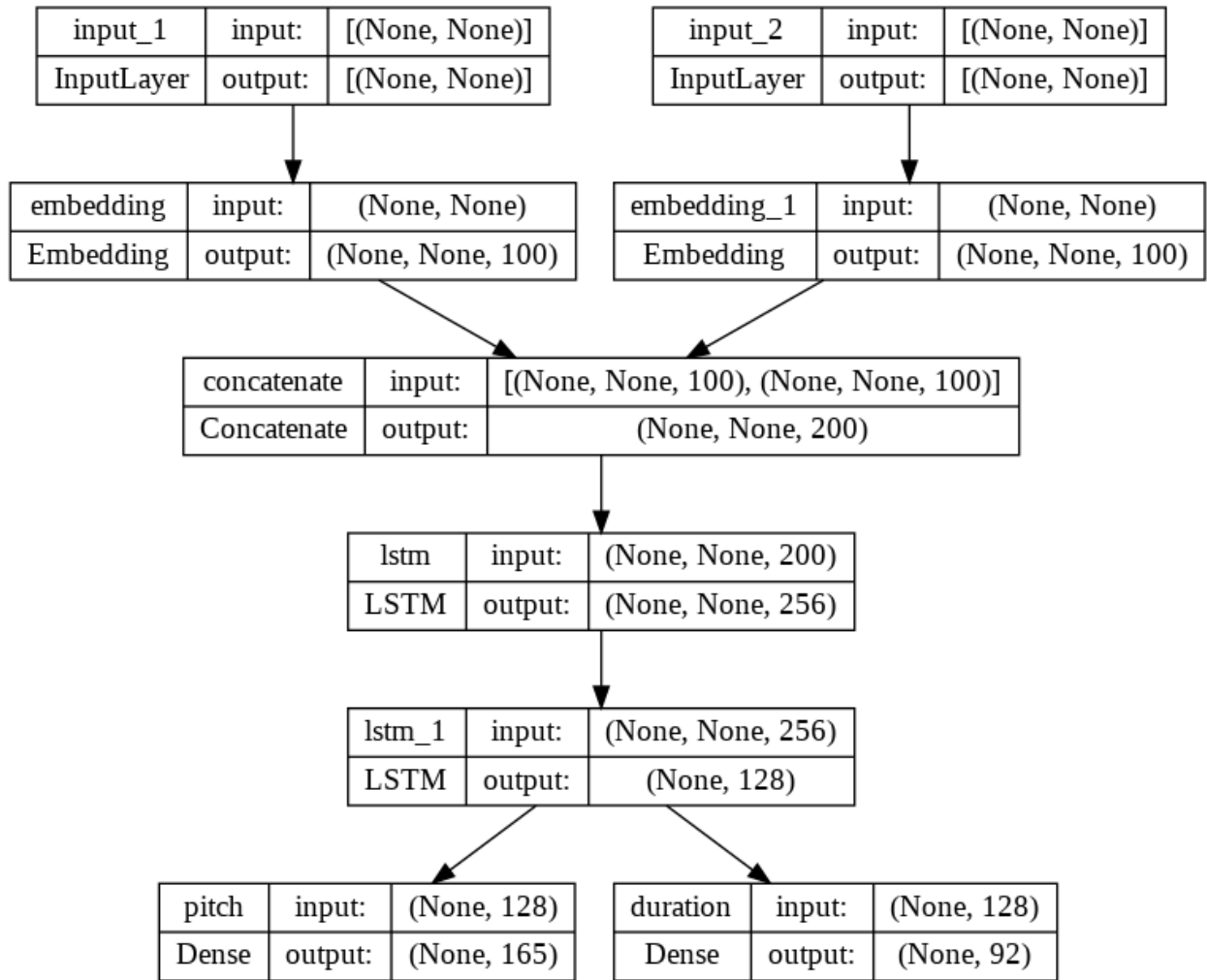


Figure 4.4: Model

Loss Tracking

Tracking of loss is the most important step in any ML-related project, which shows how the model behaves while training data is visualized using a Loss Graph. The loss graph is a plot that shows the value of the loss function over time during the training of a machine learning model. The model always tries to minimize loss.

The loss function used in the training model is KL-Divergence. KL-divergence is used to measure the difference between two probability distributions. The brief explanation for loss function KL-divergence used in the model is discussed above. The loss graph of this model while training ESAC Folk database[1] is shown below:

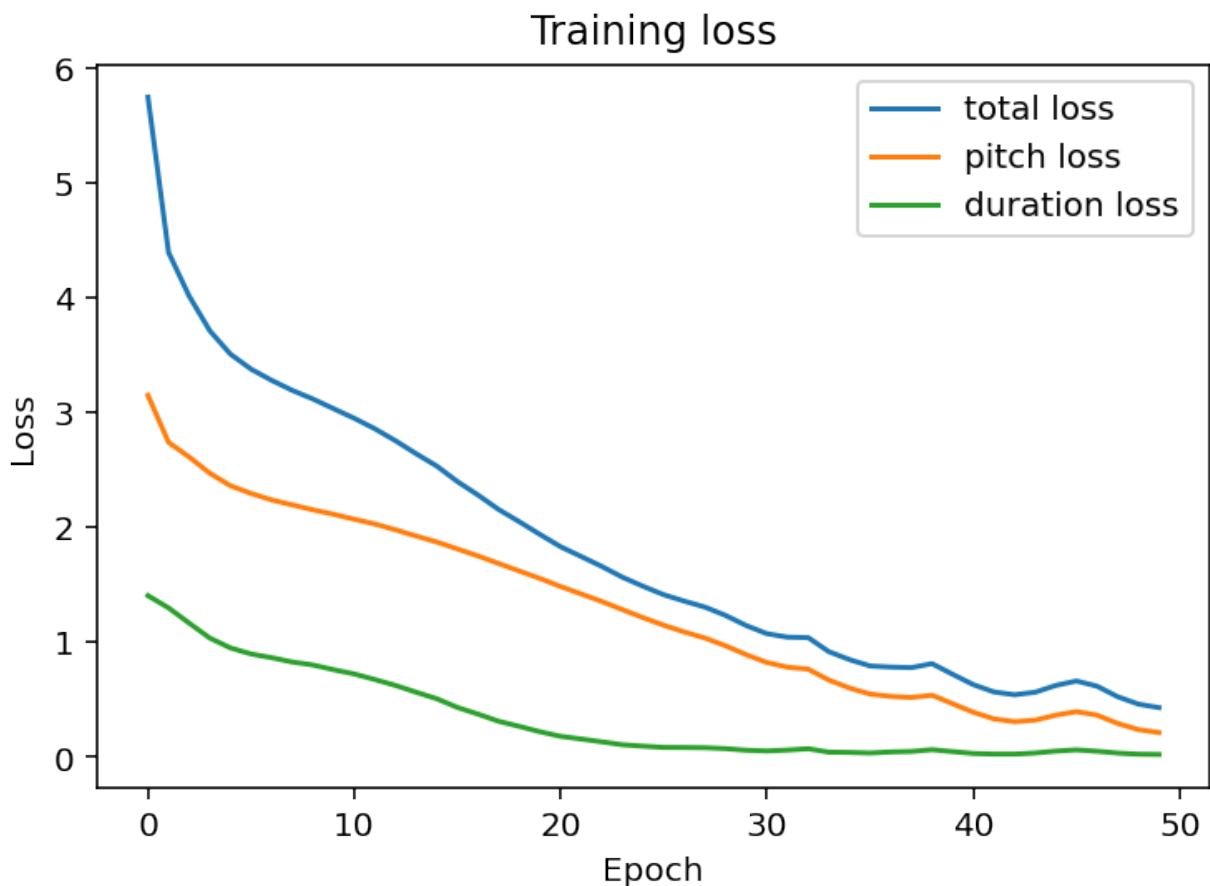


Figure 4.5: Training loss

As the model was trained using notes and duration, its respective loss and total loss are shown in (Figure 4.5). The total loss is the sum of the KL divergences calculated between the predicted output distribution and the target distribution over all samples in the training set. The model is then trained to minimize this total loss over multiple iterations. The figure shows that the model is trained well as loss converges.

Tune hyper-parameter

Tuning the hyper-parameters of a model was done to increase the performance of the model. The loss function was tracked and observed for the tuning parameter and tested generated output. The tuned parameter in the LSTM model was several hidden layers and loss function by watching the training loss and validation loss graph. Through repeated experiments, it was found adding one or two layers doesn't change loss much and the model converges quickly with high loss. So the neuron and layers used in this model are optimum for doing an experiment on a small scale as doing

research in ML depends on computing power and the size of the dataset to work with.

While working with our custom dataset, we faced difficulties due to the large number of generated tokens, which led us to introduce L1 regularization for the notes output. As our custom dataset was diverse yet relatively small, the generated tokens were substantial, and L1 regularization helped optimize the training process by driving the weights of irrelevant or minimally relevant features to zero. For the duration output, we used L1L2 regularization, where the L1 part of the regularization helped in feature selection, while the L2 part helped in reducing the model's variance.

While working with ESAC dataset[1], which had a greater volume of data and fewer tokens, we found that regularization was unnecessary as the model was able to effectively learn the dependency patterns without it.

4.1.5 Music generation

After training the model, it can be used to generate new music. For music generation, we need to provide a seed input to the model. We provided a random seed to the model, and it predicts the follow-up notes and duration based on the patterns it had learned during training. The output generated by the model is in encoded form, which is then decoded into known notes and durations. The decoded output is then converted into a music21 stream object and saved as a 'midi' file.

The music generation program provided additional options for users to control the output music. Users were able to pass in the time signature and bpm as parameters, which allowed them to control the rhythm of the music generated by the program. In addition, users were also able to select the instrument that they wanted to use to play the music. These options gave users a degree of control over the style and sound of the generated music.

4.1.6 Evaluation

The generated was then evaluated to assess its quality and determine whether it meets the desired criteria. It was mainly done by listening to music and getting feedback from musicians. This can provide valuable insights into the overall sound and feel of the music and can help identify any areas where the generated music may be lacking or could be improved.

4.2 Algorithmic Approach

The block diagram for music generation using an algorithmic approach is shown in fig (4.6).

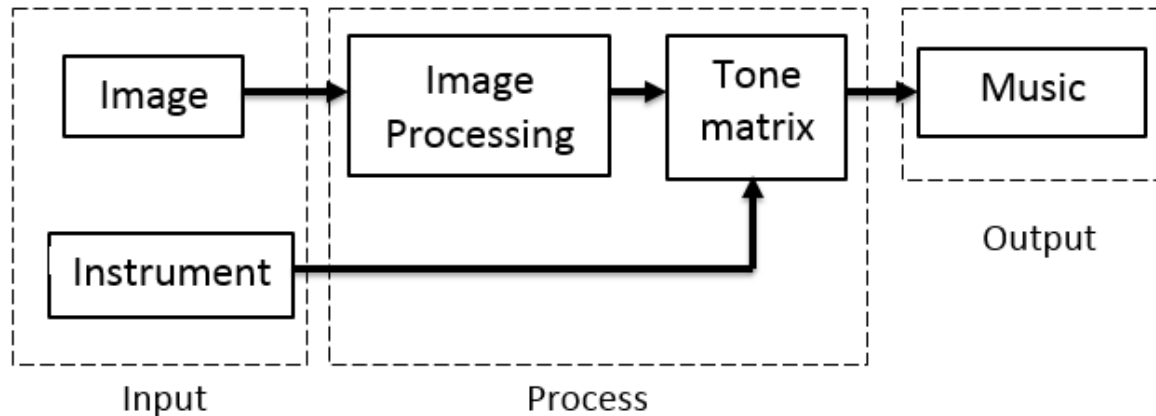


Figure 4.6: Block Diagram for music generation using algorithmic approach

The various steps involved in music generation using an algorithmic approach are discussed below:

1. **Input:** The algorithmic approach used in this project for generating music relies on image processing and a tone matrix. The processed image serves as a trigger to determine which note is played by the tone matrix. Additionally, metadata specifying the instrument is provided to select the appropriate instrument for playing the music.
2. **Image Processing:** To reduce the dimensionality of the input image, it is first converted into grayscale. The grayscale image is then resized into a 16x16 pixel image.

The resulting image is then converted into two distinct colors, black and white. The black and white image of the 16x16 matrix is used as a switch for the tone matrix. The white color represents the ON switch, and the black color represents the switch is OFF.

There are two methods used to change the grayscale image to ON and OFF switches.

- a **Max-intensity method:** Max-intensity method is used as the default method to generate monophonic music. In this method, the grayscale image is checked column-wise, and maximum intensity in that column is made ON and the rest are made OFF. If there is a presence of more than 3 pixels with the same intensity as the max intensity in the column, then only three switch positions were chosen for the ON state and others in the OFF state. This was done to avoid noise in generating music as more than 3 notes played at the same time didn't sound good.
- b **Min-intensity method:** The Min-intensity method is similar to the max-intensity method, but instead of maximum intensity, the minimum intensity was chosen to make a switch

to the ON state, and the rest were in the OFF state. Like the max-intensity method, this method also limits the 3 ON state of the switch in a column. This method was developed to generate a second musical instrument in an Algorithmic approach.

3. **Tone Matrix:** The basic concept of tone matrix used in this project is discussed above in the Related Theory section. The tone matrix has the time axis on the horizontal and frequency on the vertical axis. While the frequency axis remains constant, the length of the time axis can be adjusted by increasing or decreasing the number of columns to create music of a specific length. The ON and OFF switches are used to map the matrix.

The length of the music is determined by the size of the image, which should typically be around $16 \times N$. Increasing the value of N can extend the duration of the music accordingly.

The processed image with the ON and OFF switches is mapped to the tone matrix. The ON switches on the same column play simultaneously. Music21 library creates a sequence of notes or chords that will be played in the MIDI file.

4. **Output:** The tone matrix generates the music based on the switch (processed image) which is turned ON. The obtained output depends on the input image and instrument selection. The generated music is short and can be played in a loop, which is pleasant to hear.

The experiment was done to produce polyphonic music. One part of the music melody using an instrument was produced by processing the maximum intensity of the image and using it as a switch for the tone matrix. While another instrument music Melody was produced in a similar manner but using the minimum intensity of the image as a switch for the tone matrix. Two melodies are combined together to get a multi-instrument effect on the final music.

4.3 AI Sync Algorithm

Our project aims to enhance the quality of AI-generated music and provide a greater degree of control and uniqueness to the music by combining AI and algorithmic approaches. While AI-generated music can be unique, it may not always be enjoyable to listen to. Algorithmic approaches, on the other hand, offer more precise control to the user but require expertise to create diverse and unique compositions.

Our research focuses on a hybrid approach that combines the strengths of both AI and algorithmic approaches. We utilize image intensity to generate a melody from a tone matrix, which is then used to control and generate an AI melody. The resulting melodies are combined to create a more pleasing and unique sound. The block diagram of the AI sync algorithmic approach is shown below:

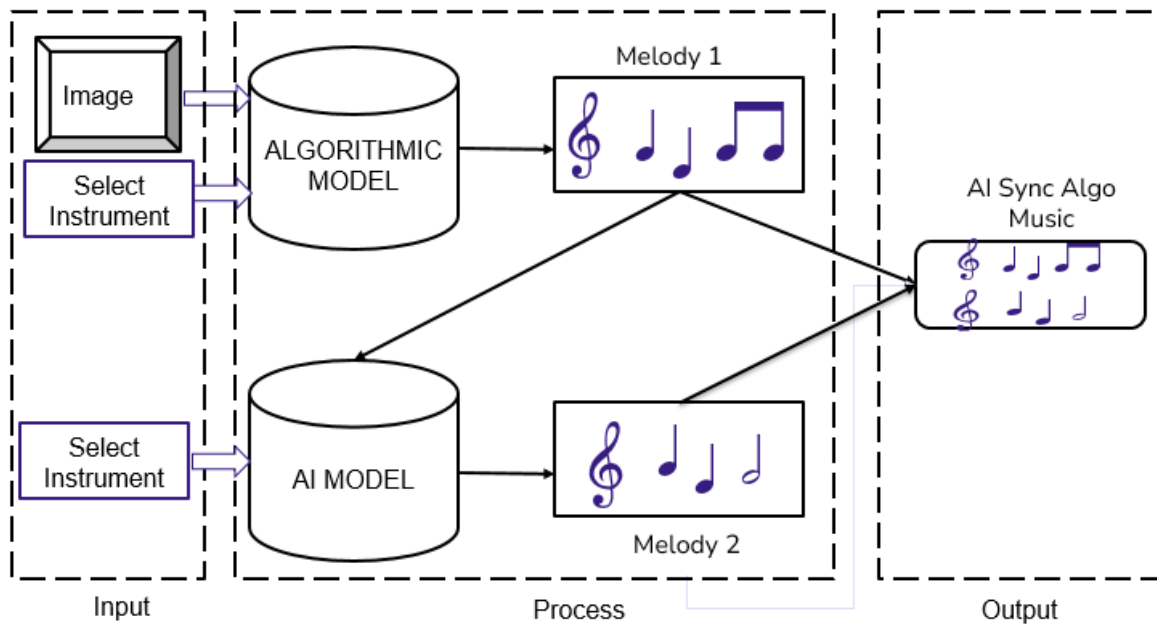


Figure 4.7: Block Diagram For AI sync Algorithm Approach

4.3.1 Working Mechanism

For synchronization of AI music with Algorithmic generated music, we first passed an input, i.e., an image to the system, which was used to generate music using the tone matrix algorithm. The notes and durations of algorithmic-generated music were extracted and passed as seeds used for generating music using AI. For the rhythm of algorithmic-generated music to sync with AI-generated music, the time signature and bpm of algorithm-generated music were also extracted and passed as metadata for constructing music from generated notes and duration from the generative model. This ensures that the AI-generated music matches the tempo and rhythm of the algorithmic-generated music. The obtained output from AI generative model and tone matrix are synchronized to get pleasant music.

4.4 UML Diagram

4.4.1 Use Case Diagram

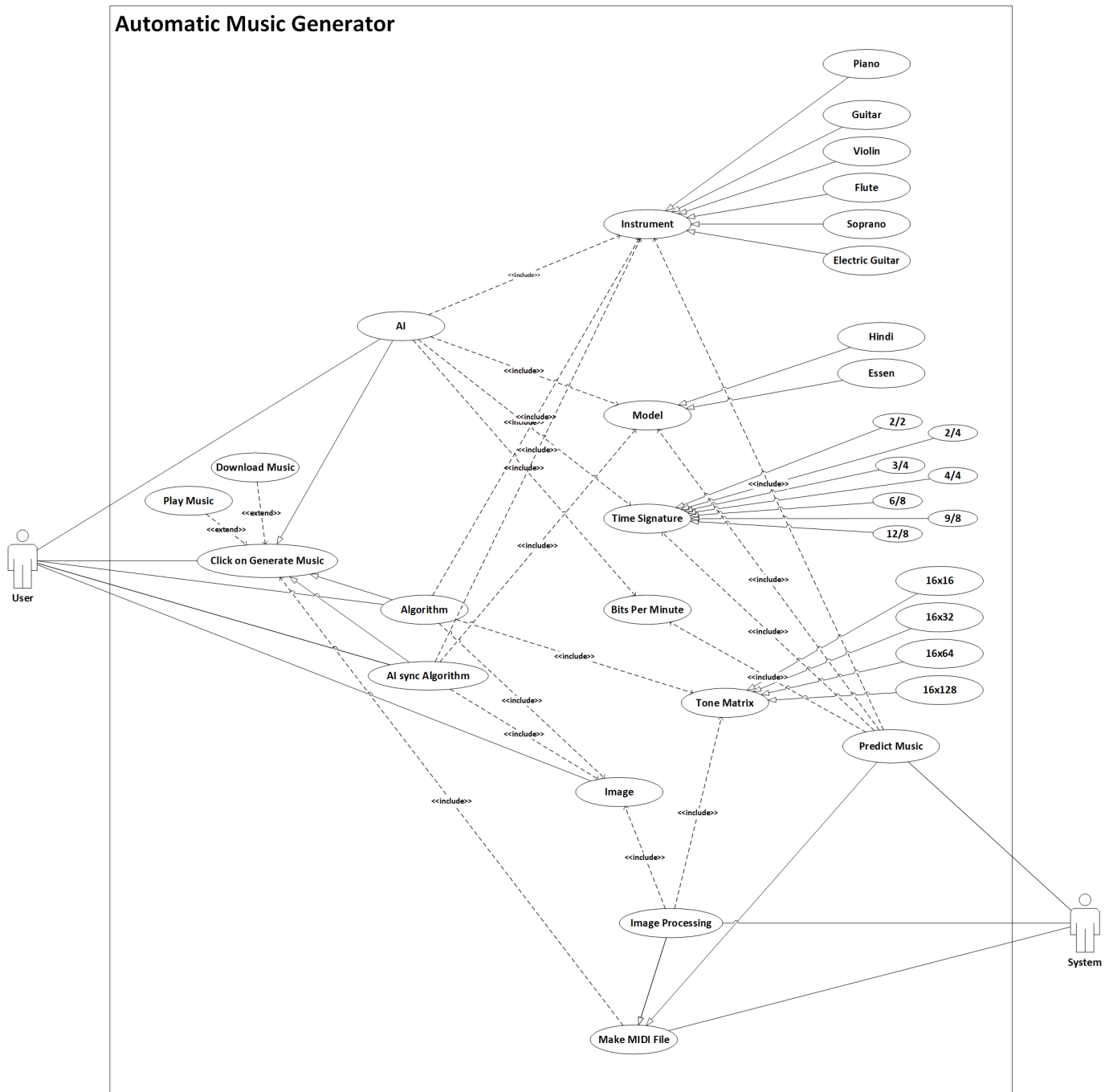


Figure 4.8: Use Case Diagram

4.4.2 Activity Diagram

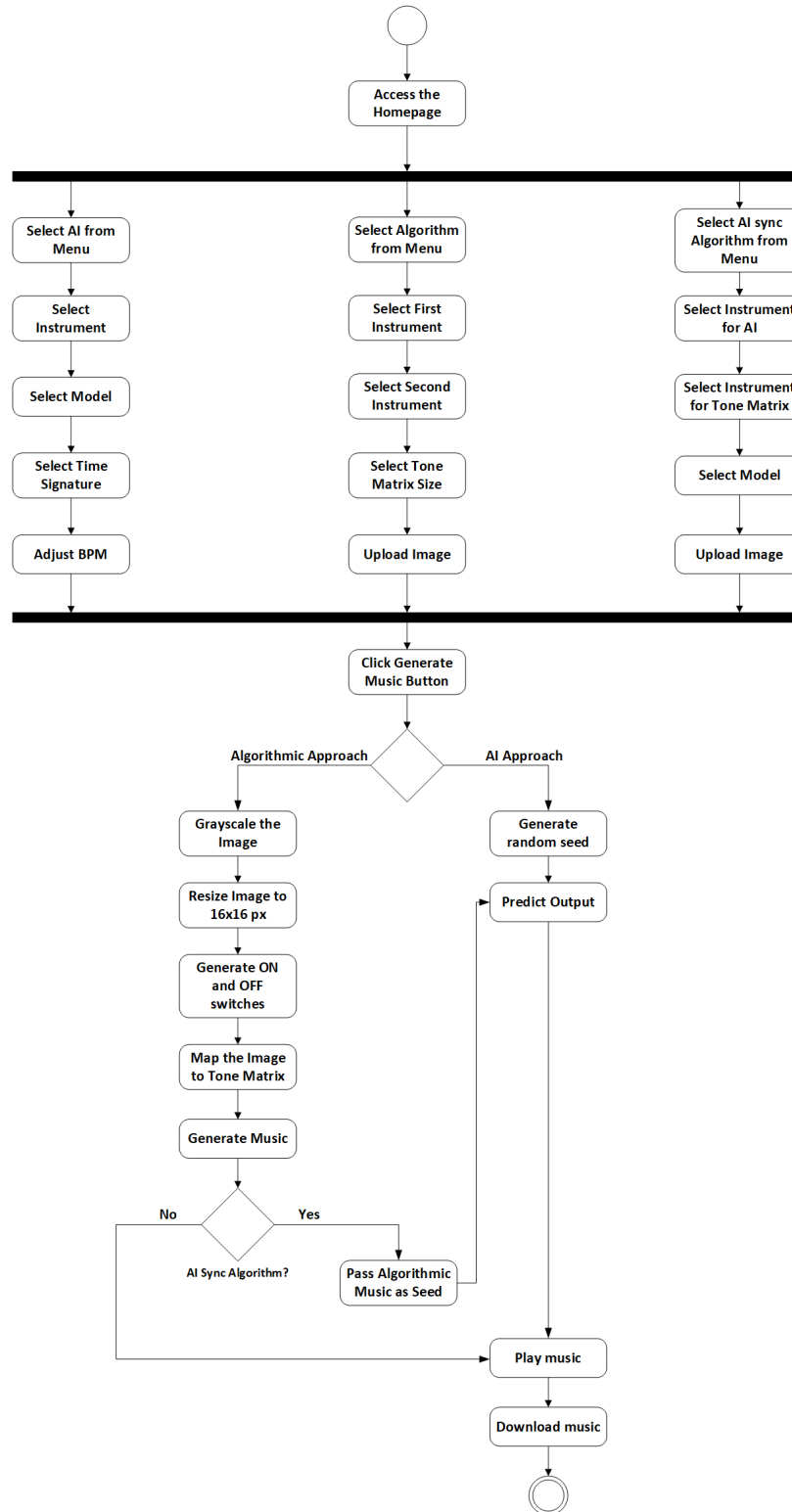


Figure 4.9: Activity Diagram

5. Experiment and Result

The LSTM was used as a generative model while working on this project. The dataset used to train the model was converted to the '.mid' file format. Notes and duration were extracted from the dataset as features to train the model. Since the model was generative, evaluation metrics used only tracked loss while training data. The loss function used in the model was KL-Divergence. Loss of model was tracked while training to visualize how the model is being trained.

While experimenting with our model with the custom dataset, we aimed to minimize loss and increase model robustness. To achieve this, we experimented with several regularization techniques such as L1, L2, and L1L2. After conducting repeated experiments with sample data, we found that L1 regularization was the most effective for our model. Notably, the loss did not converge when regularization was not utilized. As a result of these experiments, we concluded that L1 regularization should be applied to the notes classification output, while L1L2 regularization should be applied to the duration output.

While working with ESAC Folk database[1] no regularization was used. The model was trained up to 100 epochs. Once the model was trained successfully, we generated music using the trained model. Parameters such as time signature, instrument type, and bpm are in control of the user in generated music. The seed given to the generative model changes every time, which is responsible for the type of music it generates. Generated music is the continuity of seed music when decoded.

The sheet music for one of the decoded seeds while experimenting is shown in Fig(5.1). The generated music produced by the model using seed of Fig(5.1) is shown in Fig(5.2). The generated music is an original composition created by the model using the given seed as a starting point.



Figure 5.1: Decoding seed in sheet-music¹



Figure 5.2: Generated music¹

We used a simplified version of the tone matrix for the algorithmic approach with slight modifications. The image was taken as input, and it was first converted into grayscale and then resized into 16x16 pixels. Then the maximum pixel intensity of the image was taken column-wise, which was then used as a switch for the 16x16 tone matrix of fig (3.3). If there was more than two maximum intensity, the priority of the switch was given to note with minimum frequency. And a maximum number of switches that could be opened in one column was 3. For the image as shown in fig (5.3) when passed as input, the switch configuration of tone matrix is given by fig (5.4). The white pixel is ON, and the black pixel is OFF. The instrument used in this generated music is the piano. The music generated by the tone matrix in sheet music is shown in fig (5.5).



Figure 5.3: Image used for algorithmic approach music generation

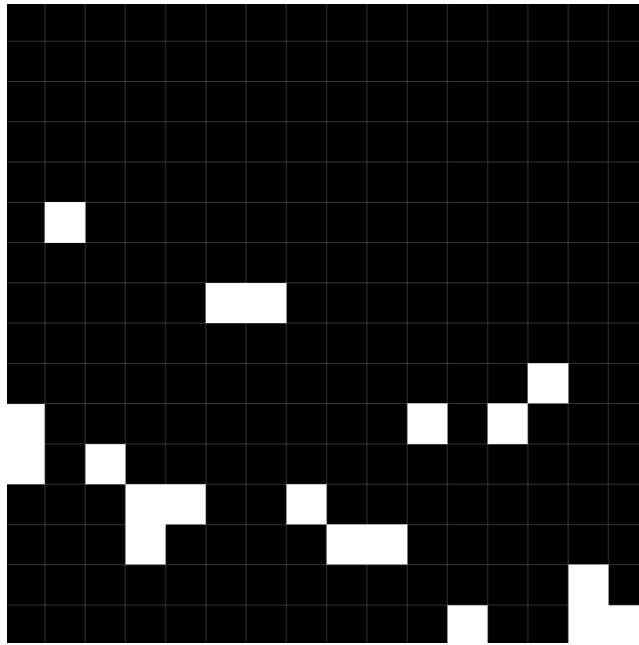


Figure 5.4: Switch configuration using maximum intensity pixel



Figure 5.5: Sheet music generated by switch of fig (5.4)

For multi-instrument music, the experiment was done with an algorithmic approach. For the second instrument music, the minimum intensity of pixel was taken. Here also, the maximum switch that can be opened in a column was set to 3. The minimum intensity switch configuration by using the image of fig (5.3) is shown in fig (5.6). The instrument used in this generated music is the flute. Its sheet music is in fig (5.7). Two music obtained from the minimum intensity and maximum intensity were configured to get the multi-instrument effect in music. The two-instrument music sheet music is shown in fig (5.8).

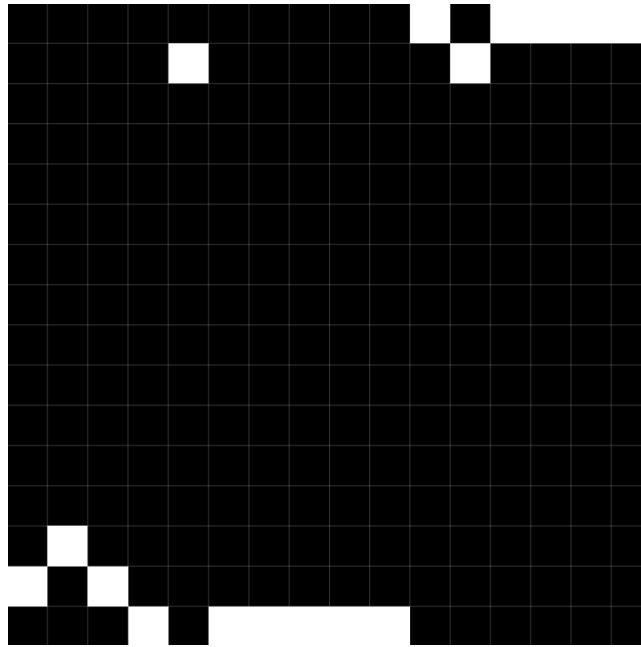


Figure 5.6: Switch configuration using minimum intensity pixel



Figure 5.7: Sheet music generated by switch of fig (5.6)



Figure 5.8: Sheet music of multi-instrument music

To synchronize AI-generated music with algorithmically generated music, we conducted several experiments. Initially, we utilized the algorithmic approach of tone matrix to create the music.

6. Output Demo

The following figures show the UI for generating music and sheet music of generated music.

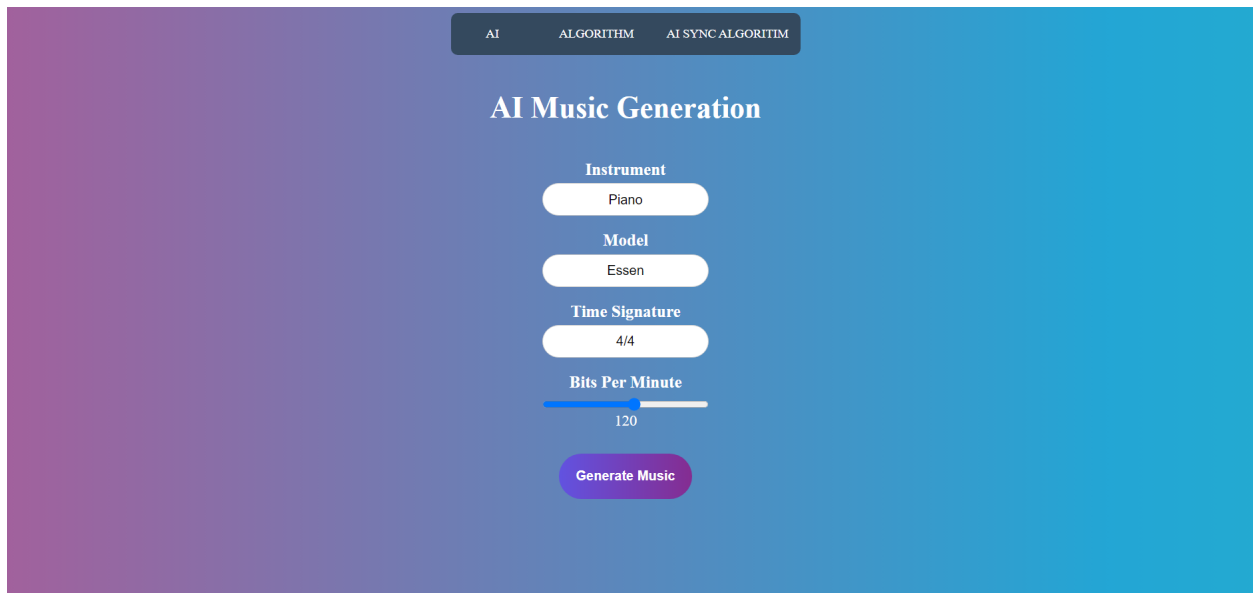


Figure 6.1: AI Music Generation Page

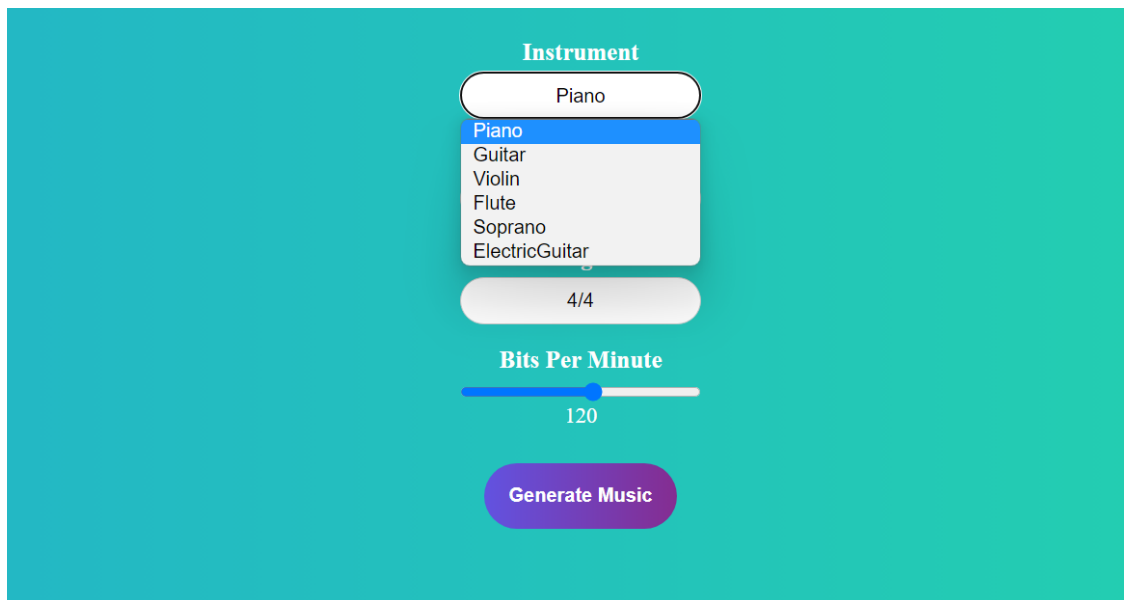


Figure 6.2: Instrument Selection

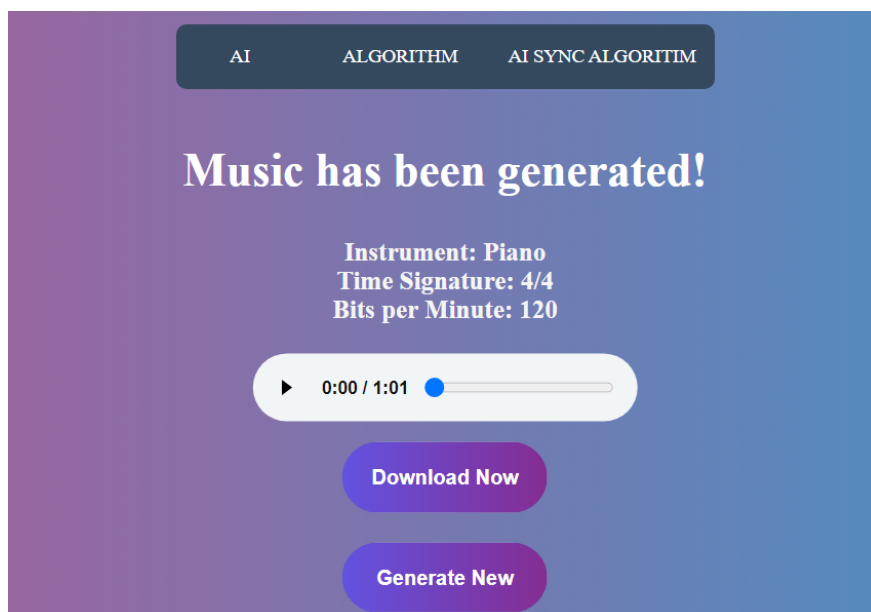


Figure 6.3: AI Music Generation Output Page



Figure 6.4: Sheet Music for AI-Generated Music

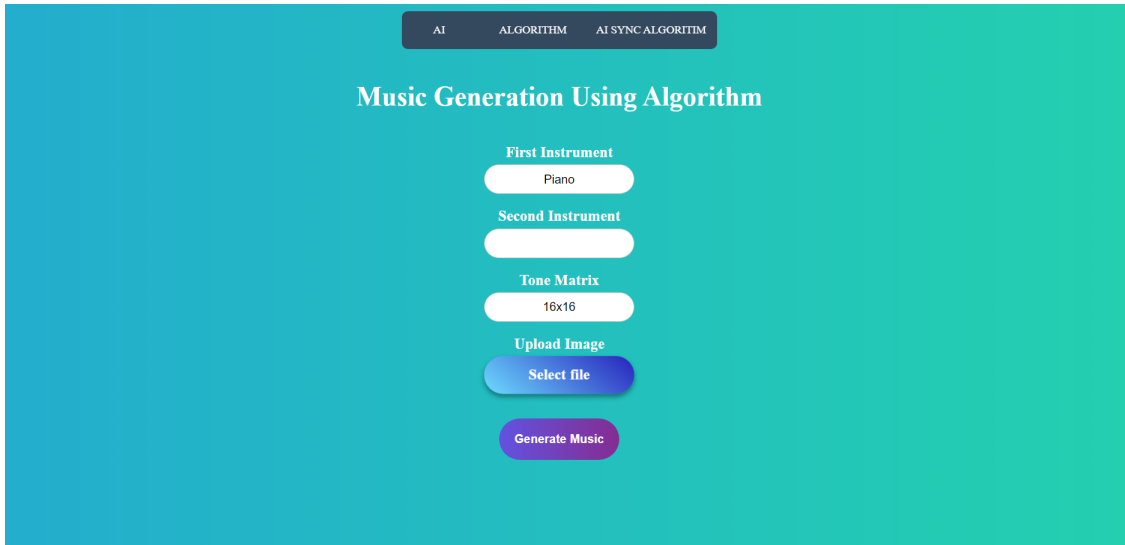


Figure 6.5: Algorithmic Music Generation Page



Figure 6.6: Algorithmic Music Generation Output Page

The image displays a sheet music score for an algorithmically generated piece. It consists of four staves. The top staff is for Piano, marked 'Piano, Piano', with a tempo of 120 beats per minute. The second staff is for Violin, labeled 'Pardessus de viole, Violin'. The third staff is for Piano accompaniment, labeled 'Pno.', and includes a triplet of eighth notes. The bottom staff is for the right hand of the piano, labeled 'Pds. v.'. The music is written in 4/4 time and features a mix of eighth and sixteenth notes, with some rests and dynamic markings.

Figure 6.7: Sheet Music for Algorithmic Generated Music

The image shows a web interface for AI music generation. At the top, there are three tabs: 'AI', 'ALGORITHM', and 'AI SYNC ALGORITHM'. The main heading is 'AI Music Generation Sync with the Algorithmic Music Rhythm'. Below this, there are several interactive elements:

- 'Instrument For AI' with a 'Piano' button.
- 'Instrument For Tone Matrix' with a 'Flute' button.
- 'Model' with an 'Essen' button.
- 'Upload Image' with a 'Select Image' button.
- A 'Generate Music' button at the bottom.

 The interface has a blue and purple gradient background.

Figure 6.8: AI Sync Algorithm Music Generation Page

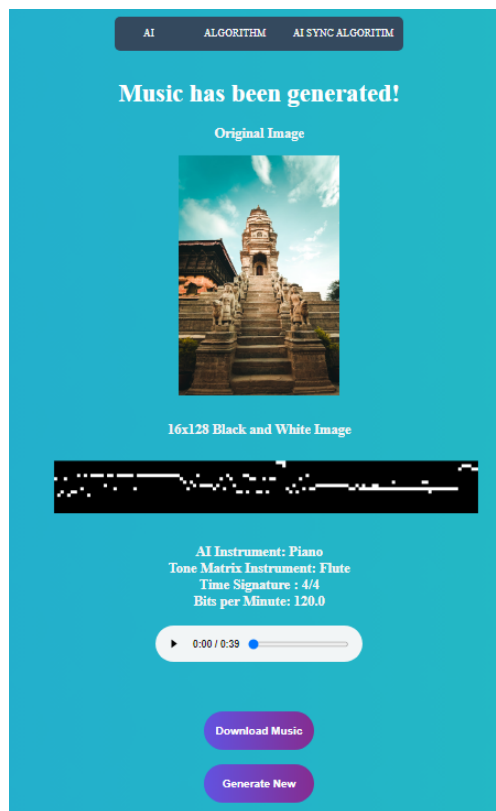


Figure 6.9: AI sync Algorithm Music Generation Output Page

Figure 6.10: Sheet Music for AI sync Algorithm Generated Music

7. Conclusion

In this project, we focused on developing machine-learning techniques for music generation that could help musicians compose and refine music interactively. Notes, chords, and their duration were taken as the core components of music for the training model. We generated notes and duration, which then was used for music reconstruction. Since many other parameters change the rhythm of how music is played. We allowed the passing of another argument, such as time signature, bpm, and instruments as user input while generating music, allowing the user to control the rhythm of the music. This brings greater flexibility to control the rhythm of the music. Combining AI-generated melody pieces with algorithmic tone matrix-generated melody pieces to produce music is more pleasing, and the degree of control of generated music is in the hands of the user.

However, our work also encountered several challenges, including insufficient high-quality, labeled music data, musical complexity, etc. Despite these challenges, we believe that our approach holds great promise for the future of the music generation, and we hope that our work will inspire further research in this area.

There are many potential directions for future work. It will likely involve new techniques like enhancing the learning process to produce more satisfying music, exploring the reinforcement learning algorithm, fusing emotion with music creation to investigate how music can influence human emotions, enhancing human-AI connection to enhance the objective of music, integrating autonomous music generating technologies into daily life and so on.

We believe these directions will lead to even greater advances in music generation, and we look forward to seeing the progress that will be made in the coming years.

References

- [1] Dr. Ewa Dahlig. ESAC Data Homepage. <http://www.esac-data.org/>, February 2011. Accessed: February 20, 2023.
- [2] Quizlet. Notes, rests, names, and values diagram, n.d. Accessed on December 22, 2021.
- [3] Randall Richard Spangler. *Rule-based analysis and generation of music*. California Institute of Technology, 1999.
- [4] Claude V Palisca and Dolores Pesce. Guido of arezzo [aretinus]. *The New Grove Dictionary of Music Online*. Retrieved November, 5:2003, 2001.
- [5] Iannis Xenakis. The origins of stochastic music 1. *Tempo*, (78):9–12, 1966.
- [6] Jiyabo Cao, Jinan Fiaidhi, and Maolin Qi. A review of automatic music generation based on performance rnn. 2020.
- [7] Haohang Zhang, Letian Xie, and Kaiyi Qi. Implement music generation with gan: A systematic review. In *2021 International Conference on Computer Engineering and Application (ICCEA)*, pages 352–355. IEEE, 2021.
- [8] Emma Frid, Celso Gomes, and Zeyu Jin. Music creation by example. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–13, 2020.
- [9] David Kang, J Kim, and Simen Ringdahl. Project milestone: Generating music with machine learning. *Stanford University, CA, USA*, 2018.
- [10] Nico Schuler. From music grammar to cognition of music in the 1980s and 1990s: the surplus history of computer-based music analysis. *Muzikoloski Zobornik*, 43, 2007.
- [11] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [12] Jean-Pierre Briot. From artificial neural networks to deep learning for music generation: history, concepts, and trends. *Neural Computing and Applications*, 33:39–65, 2021.

- [13] Isaac Tham. Generating music using deep learning. <https://towardsdatascience.com/generating-music-using-deep-learning-cb5843a9d55e>, Aug 2021.
- [14] Douglas Eck. Welcome to magenta! <https://magenta.tensorflow.org/blog/2016/06/01/welcome-to-magenta/>, Jun 2016.
- [15] Adam Roberts, Curtis Hawthorne, and Ian Simon. Magenta.js: a javascript api for augmenting creativity with deep learning. 2018.
- [16] Ji-Sung Kim. Deep learning-driven jazz generation. <https://deepjazz.io/>.
- [17] Harun Zulic. How ai can change/improve/influence music composition, performance and education: Three case studies. *INSAM Journal of Contemporary Music, Art and Technology*, 2019.
- [18] Heewoo Prafulla Dhariwal and JunChristine McLeavey Payne. Jukebox. <https://openai.com/blog/jukebox/>, Jun 2021.
- [19] Mingliang Zeng, Xu Tan, Rui Wang, Zeqian Ju, Tao Qin, and Tie-Yan Liu. Musicbert: Symbolic music understanding with large-scale pre-training. *arXiv preprint arXiv:2106.05630*, 2021.
- [20] NA Nikitin, VL Rozaliev, Yu A Orlova, and AA Alekseev. Automated sound generation based on image color spectrum with using the recurrent neural network. In *CEUR Workshop Proceedings*, volume 2212, pages 399–408, 2018.
- [21] Elena Rivas Ruzafa. *Pix2Pitch: generating music from paintings by using conditionals GANs*. PhD thesis, ETSI_Informatica, 2020.
- [22] R Madhok, S Goel, and S Garg. Sentimozart: Music generation based on emotions. *icaart 2018-proceedings of the 10th international conference on agents and artificial intelligence, 2 (icaart)*, 501–506, 2018.
- [23] Xiaodong Tan, Mathis Antony, and H Kong. Automated music generation for visual art through emotion. In *ICCC*, pages 247–250, 2020.
- [24] Sean Vasquez and Mike Lewis. Melnet: A generative model for audio in the frequency domain. *arXiv preprint arXiv:1906.01083*, 2019.
- [25] Huanru Henry Mao, Taylor Shin, and Garrison Cottrell. Deepj: Style-specific music generation. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 377–382. IEEE, 2018.

- [26] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*, 2017.
- [27] François Pachet. Automatic generation of music with recurrent neural networks. In *Proceedings of the 20th European conference on artificial intelligence*, pages 411–416. IOS Press, 2013.
- [28] Gaëtan Hadjeres and François Pachet. Music generation with neural networks. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 180–188, 2016.
- [29] Joseph Rothstein. *MIDI: A comprehensive introduction*, volume 7. AR Editions, Inc., 1992.
- [30] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation—a survey. *arXiv preprint arXiv:1709.01620*, 2017.
- [31] Pluralsight. Introduction to lstm units in rnn, n.d. Accessed on December 22, 2021.
- [32] H David. von seggern. crc standard curves and surfaces with mathematica. *H. von Seggern David*, 2016.
- [33] Wolfram Research. Tanh. <https://reference.wolfram.com/language/ref/Tanh.html>, 2021.
- [34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [35] Solomon Kullback and Richard A Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.
- [36] Deepa. Ridge-regression-lasso-regression-elastic-net-regression. <https://learnerjoy.com/ridge-regression-lasso-regression-elastic-net-regression/>, March 5 2022. [Online; accessed 5-March-2023].