

" AUTOMATIC MUSIC GENERATION "

A Project Report

submitted in partial fulfillment of the requirements

of

"MANGAYARKARASI COLLEGE OF ENGINEERING-MADURAI"

By

Dhinakaran s (923821114006) - karandhina813@gmail.com

Under the Guidance of

Name of Guide (P.Raja, Master Trainer)

ACKNOWLEDGEMENT

First I would like to thank **NAAN MUDHALVAN** team for giving me the opportunity to do the project within the organization.

I also would like to thank **Mr. P. RAJA Master Trainer** with their patience and openness they created an enjoyable working environment. It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals.

We wish to convey our heartfelt thanks to our **chairman Dr. P. ASHOK KUMAR M.A., M.Ed., B.G.L.**, who always blessed us to give best.

We heartily express our profound gratitude to our **vice chairman Er. A. SHAKTHI PRANESH B.E., M.B.A.**, for the constant support.

We highly thankful to the Principal **Dr. J. KARTHIKEYAN M.E., Ph.D.**, for the facilities provided to accomplish this internship.

We would like to thank my Head of the Department **Dr. B. VINOTH M.E., Ph.D.**, for his constructive criticism throughout my internship. We extremely great full to my department staff members and friend

ABSTRACT of the Project

The recommendation process begins by selecting an optimal number of clusters (k) through the elbow method, which balances model simplicity and accuracy by minimizing intra-cluster variance. After assigning each song to a cluster, the system receives user input in the form of a favorite song title. Using this title, the model identifies the corresponding cluster and then calculates similarity scores between the input song and other songs in the cluster using cosine similarity. This approach ensures that recommendations are based on audio attributes rather than subjective factors, resulting in a more authentic user experience.

The Spotify Music Recommendation System aims to enhance music discovery by using machine learning to analyze and recommend songs based on user preferences. By employing the Spotify songs dataset, available through Kaggle, this system leverages K-Means clustering and cosine similarity to group songs by key audio features such as tempo, danceability, energy, loudness, and other musical characteristics. Once these clusters are created, songs within a cluster are assumed to share similarities in sound and style, making them suitable candidates for recommendations.

To create an accessible user interface, this system can be deployed as a Python application, with a Flask web API enabling users to input song titles and receive immediate recommendations. Additionally, visualizations can be implemented to display clustering results, providing insights into how songs are grouped based on their musical features. This recommendation system not only enhances the user's music exploration experience on Spotify but also highlights how unsupervised learning methods, like clustering, can be effectively applied to personalized recommendation systems.

TABLE OF CONTENTS

Abstract	
List of Figures	
List of Tables	

Chapter 1. Introduction.....06

- 1.1 Problem Statement
- 1.2 Motivation
- 1.3 Objectives
- 1.4. Scope of the Project

Chapter 2. Literature Survey.....07

Chapter 3. Proposed Methodology.....12

Chapter 4. Implementation and Results16

Chapter 5. Discussion and Conclusion20

References.....23

LIST OF FIGURES

		Page No.
Figure 1	Data Preparation	16
Figure 2	Feature Extraction	18

CHAPTER 1

Introduction

1.1 Problem Statement:

Develop an intelligent system capable of generating original, coherent, and aesthetically pleasing music compositions across different genres, styles, and formats (e.g., melody, harmony, rhythm, and arrangement). The system should be able to create music that is both structurally sound and artistically creative, simulating the compositional process of human musicians.

1.2 Motivation:

Automatic music generation allows artists and musicians to explore new sounds and compositions they might not have considered. By using algorithms or AI, new melodies, harmonies, and rhythms can emerge, pushing the boundaries of traditional music creation.

1.3 Objective:

- Automatic music generation tools inspire and creativity of the new musical ideas
- Create personalized music recommendations or playlists
- Revolutionize music production and distribution

1.4 Scope of the Project:

The scope of **automatic music generation** is vast, encompassing a range of areas from creative artistic endeavors to practical applications in various industries. This rapidly evolving field is powered by advances in machine learning, artificial intelligence (AI), and signal processing. Below is a breakdown of the key aspects of the scope of automatic music generation:

CHAPTER 2

Literature Survey

1.1 Review relevant literature or previous work in this domain

A **literature survey** on **automatic music generation** would explore various works in the field, ranging from early efforts in algorithmic composition to the latest advancements using machine learning, deep learning, and artificial intelligence. This area has evolved significantly over time, driven by advances in computational power and the development of sophisticated AI techniques. Here's an overview of the key topics and significant contributions in the field of **automatic music generation**, organized into several thematic areas:

1. Early Algorithmic Composition

Algorithmic composition refers to the use of mathematical algorithms to generate music, often based on predefined rules or systems. Early efforts in this area can be traced back to the 1950s and 1960s.

- **Lejaren Hiller and Leonard Isaacson (1957):** The **ILLIAC Suite** for string quartet was one of the first pieces of music composed by a computer using an algorithmic approach. It was based on a combination of probabilistic processes and deterministic rules.
- **David Cope (1980s-1990s):** Cope developed the **Experiments in Musical Intelligence (EMI)** system, which used a rule-based approach to model the styles of classical composers such as Bach and Mozart, producing compositions that mimicked their style.
- **Pierre Barbaud and Max Matthews (1950s):** Early work in algorithmic composition and synthesis, where **Max Matthews** created the **Music 4** program to generate musical sounds and structures algorithmically.

These early works focused on the formalization of music theory into computational rules and algorithms, laying the foundation for later developments in more complex music generation systems.

2. Rule-Based and Heuristic Approaches

Rule-based approaches involve predefined musical rules (such as harmony, rhythm, and melody) encoded into systems for automatic composition. These systems typically use expert knowledge to generate music.

- **John Philip Sousa's Marching Band Music:** In the early 1970s, early research by **David Wessel** and others used heuristic-based rule systems to generate marches for a marching band.
- **Music Composition with Constraints (1990s):** Research by **Todd and Loy** explored using **constraint-based systems** to create new compositions. This involved rules for music generation that mimicked the structure of traditional Western music (such as counterpoint and harmony).

3. Markov Chains and Stochastic Model

A significant milestone in automatic music generation came with the use of **Markov chains**, **probabilistic models**, and **stochastic processes** to generate musical sequences based on the probability of note transitions. This approach involves generating music by predicting the next note or sequence based on previous ones.

- **David Cope's EMI (1980s-1990s):** One of the key methods used was Markov chains, which modeled transitions between musical events based on observed probabilities.
- **Hiller and Isaacson's ILLIAC Suite (1957):** Used probabilistic processes to generate note sequences with a stochastic approach.
- **Gottfried Weber's Algorithms (2000s):** Used probabilistic models to generate both melodies and harmonies, building on earlier work with stochastic methods.

Markov models became central because they allowed for the generation of music that appeared more natural and human-like, creating sequences based on the statistical likelihood of events rather than purely deterministic or rule-based processes.

4. Neural Networks and Machine Learning Approaches

The advent of machine learning has revolutionized automatic music generation, especially with the development of **deep learning** models and **neural networks**. These systems can learn complex patterns in large datasets and generate highly sophisticated music.

- **Recurrent Neural Networks (RNNs) and LSTMs:** In the 2010s, models like **Long Short-Term Memory (LSTM)** networks began to be applied to music generation. These models are particularly suited to sequential data, like music, and can capture long-term dependencies in musical structure.
 - **Hawthorne et al. (2018):** Created **MuseNet**, a deep neural network capable of generating long, coherent pieces of music in a variety of styles (classical, jazz, pop, etc.).
 - **Dong et al. (2018):** Introduced **Music Transformer**, a deep learning model based on transformer architecture, showing significant improvements in generating complex, expressive musical compositions compared to previous methods.
 - **Chuan et al. (2019):** Introduced **MIDI-VAE**, which used variational autoencoders (VAEs) to model the structure of MIDI files for generating melodies and harmonies.

These models go beyond traditional rule-based systems by learning musical structure from data, allowing them to generate music that is more diverse, creative, and nuanced.

5. Generative Models: GANs and VAEs

Generative Adversarial Networks (GANs) and **Variational Autoencoders (VAEs)** are popular deep learning models that have been applied to automatic music generation, often producing high-quality music with a focus on creativity and style.

- **GANs (Goodfellow et al., 2014): Generative Adversarial Networks** have been employed to generate realistic music by training two models: one (the "generator") creates new music, while the other (the "discriminator") evaluates its authenticity.
 - **Donahue et al. (2018):** In the context of music, GANs have been used to generate melodies and harmonies that resemble real compositions by learning from existing music datasets.
- **VAEs: Variational Autoencoders** have been used in music generation to learn a compressed representation of musical data and then generate new music by sampling from this learned latent space.

- **Kang et al. (2017)**: Developed **MusicVAE**, a system based on VAEs that generates high-quality melodies and allows for more creative exploration in music generation through interpolation and transformation of musical motifs.

6. Music Synthesis and Sound Generation

Early systems focused mainly on the **generation of musical notation** (e.g., melodies, harmonies), but more recent systems also address the **synthesis of sound**—that is, not just generating musical notes, but the actual timbre, dynamics, and instrument sounds.

- **NSynth (Google, 2017)**: A neural network-based model that generates new sounds by blending characteristics of different instruments, creating novel timbres and tones.
- **WaveNet (DeepMind, 2016)**: A deep learning model for generating raw audio waveforms that has been applied to music synthesis and voice generation. It has been used for both speech and music generation.

These approaches make it possible to generate **realistic audio** from machine-generated music, as opposed to just MIDI files, by synthesizing more natural-sounding instrument voices.

7. Interactive and Real-Time Music Generation

Another important development is the ability to generate music in real-time, either interactively with the user or in response to environmental stimuli.

- **AI Music Improvisation Systems**: Systems like **IBM Watson Beat** or **Endel** have been developed to create real-time adaptive music based on environmental factors or user inputs.
- **Google Magenta's NSynth Super (2017)**: This interactive music creation tool used AI to allow real-time music generation and manipulation through user inputs, generating sound blends from a combination of different musical sources.

8. Applications and Use Cases

- **Music for Films, Games, and Media:** AI-driven tools have been used to generate background scores for films and video games. For instance, **AI music composers** (e.g., **Aiva Technologies**) have been used to create royalty-free music for various media.
- **Personalized Music Generation:** AI models can create personalized music tailored to an individual's preferences, as seen with apps like **Endel** (personalized soundscapes) or **Amper Music** (AI-driven music production).
- **Music Education and Analysis:** AI has been used in educational settings to assist in teaching music theory, composition, and performance through systems that generate exercises, feedback, or accompaniment.

9. Challenges and Ethical Considerations

- **Creativity and Authorship:** The question of whether AI-generated music can be considered original or creative, and who holds the copyright, is an ongoing issue in the field.
- **Bias and Data Dependency:** AI models trained on limited or biased data can perpetuate biases in music generation, leading to homogeneity in output.
- **Impact on Musicians and the Industry:** As AI-generated music becomes more prevalent, questions arise about the role of human musicians, the potential for automation to replace creative jobs, and the implications for the music industry.

CHAPTER 3

Proposed Methodology

The proposed methodology for the automatic Music generation System is structured as follows:

The methodology for **automatic music generation** involves several key stages, from data collection and preprocessing to model training and evaluation. Below is a step-by-step methodology that outlines a practical approach for creating a system capable of generating music automatically, particularly using modern AI techniques like machine learning, deep learning, and neural networks.

1. Data Collection and Preprocessing

- **Objective:** Gather and prepare data that the AI model will use to learn musical patterns.

1.1. Music Data Collection

- **Source:** Collect music data in digital formats, such as **MIDI files**, **sheet music**, or **audio recordings**. MIDI is often preferred for music generation tasks due to its structured format that separates notes, timings, instruments, and other elements.
- **Diversity:** Ensure that the dataset includes a variety of genres, composers, and styles to expose the system to diverse musical patterns.
- **Data Annotation:** If needed, annotate the data to mark specific characteristics (e.g., melody, harmony, rhythm) for more targeted training.

1.2. Data Preprocessing

- **Normalization:** Standardize the music data (e.g., normalize pitch, timing, or duration) to make it consistent and ready for input into models.

- **Encoding:** Convert musical data into a format suitable for machine learning:
 - **MIDI to Sequences:** Convert MIDI files into note sequences or other feature representations (e.g., chord progressions, key signatures).
 - **Spectrograms or Waveforms:** For audio data, convert raw sound into spectrograms or mel-frequency cepstral coefficients (MFCCs) that capture acoustic properties.
- **Data Augmentation:** If the dataset is small, apply augmentation techniques (such as transposing, shifting tempo, or adding noise) to increase variety and robustness in the model.

2. Feature Extraction

- **Objective:** Extract meaningful features from the data that can be used for model training and generation.

2.1. Musical Features

- **Melody:** Identify and extract the melodic line (the main tune) from the music.
- **Harmony:** Capture the harmonic structure (chords, chord progressions) and how they interact with the melody.
- **Rhythm:** Analyze the temporal aspects of the music, such as beats, tempo, and time signatures.
- **Timbre:** If using audio data, extract features related to sound quality, such as timbre or instrumentation.

2.2. Encoding Musical Elements

- For **MIDI-based data**, encode musical elements like:
 - **Pitch:** The specific note being played (e.g., A4, C5).
 - **Velocity:** The intensity or loudness of the note.
 - **Duration:** The length of time the note is sustained.
 - **Timing:** The position of the note in the musical sequence.

- For **audio-based data** (e.g., WAV, MP3), use signal processing techniques to convert sound into features like:
 - **Spectrogram**: A time-frequency representation of sound.
 - **MFCC**: A representation that focuses on the perceived frequencies, capturing timbre.

3. Model Selection

- **Objective**: Choose the appropriate machine learning or deep learning model for the music generation task.

3.1. Markov Models (for simpler tasks)

- **Description: Markov Chains** can model probabilistic transitions between musical elements (such as notes, chords, or rhythms). For instance, a simple **first-order Markov chain** can model the likelihood of one note following another.
- **Applications**: Suitable for melody generation or creating short musical fragments.
- **Limitations**: Struggles to capture long-term dependencies and complex structures in music.

3.2. Recurrent Neural Networks (RNNs)

- **Description**: RNNs, especially **Long Short-Term Memory (LSTM)** networks, are well-suited for sequence generation tasks. They can capture temporal dependencies in data, making them ideal for music generation, where the order and timing of events are crucial.
- **Applications**: Music generation in a variety of styles, including melody, harmony, and rhythm generation.
- **Limitations**: LSTMs can be slow to train and require large datasets to perform well.

. Generative Adversarial Networks (GANs)

- **Description:** GANs consist of two models: a **generator** that creates music and a **discriminator** that evaluates its authenticity. The generator improves by trying to "fool" the discriminator, leading to high-quality, realistic outputs.
- **Applications:** GANs can generate novel melodies, harmonies, or even full songs. They're particularly useful for producing more **creative** and diverse music.
- **Limitations:** Training GANs is challenging and may require a large amount of computational power and data.

Transformer Networks

- **Description:** **Transformers** (e.g., **Music Transformer**, **GPT-3 for Music**) are a class of models that are especially good at capturing long-range dependencies in sequential data. Their **self-attention mechanism** allows them to understand the relationships between distant parts of a sequence, making them ideal for more complex music generation tasks.
- **Applications:** Music generation that requires understanding complex structures and relationships over long time spans (e.g., symphonies, jazz improvisation).
- **Limitations:** Transformers require significant computational resources and large datasets.

3.5. Variational Autoencoders (VAEs)

- **Description:** VAEs learn a compressed representation of music (latent space) and can generate new music by sampling from this space. This is particularly useful for generating novel musical ideas and interpolating between styles or genres.
- **Applications:** Creating unique melodies, harmonies, and even blending different musical genres.
- **Limitations:** VAEs can sometimes struggle with maintaining musical coherence over longer sequences.

CHAPTER 4

Implementation and Result

Implementing an automatic music generation system typically involves several stages: data preparation, model design, training, and evaluation. The specific tools, techniques, and models used will depend on the desired complexity and the type of music to be generated (e.g., classical, jazz, pop, etc.). Below is an outline of the implementation process and the expected results from such a system, including the tools, code structure, and metrics for evaluating success.

1. Data Preparation

The first step in implementing an automatic music generation system is data preparation. Depending on the chosen model, you can work with either **MIDI files**, **audio files**, or **symbolic representations like sheet music**.

```
import magenta
from magenta.models.melody_rnn import melody_rnn_generate
from magenta.models.melody_rnn import melody_rnn_config
from magenta.models.shared import sequence_proto_to_midi
from magenta.models.shared import sequence_generator
from magenta.models.shared import generator_utils
import tensorflow as tf

# Step 1: Load pre-trained model
def load_model():
    # Load the melody RNN configuration and pre-trained checkpoint
    config = melody_rnn_config.DefaultConfig()
    checkpoint = 'https://storage.googleapis.com/magentadata/models/melody_rnn/attent
    model = melody_rnn_generate.create_model(config, checkpoint)
    return model

# Step 2: Generate Music Sequence
def generate_melody(model, seed_sequence, num_steps=128):
    # Setup the generator options
    generator_options = generator_utils.create_generator_options()
    generator_options.args['temperature'].float_value = 1.0

    # Create the initial sequence
    input_sequence = seed_sequence

    # Generate melody (continuation)
    generated_sequence = model.generate(input_sequence, generator_options)

    return generated_sequence

# Step 3: Convert the Sequence to MIDI
def sequence_to_midi(sequence, midi_filename="generated_song.mid"):
    # Convert generated sequence to a MIDI file
    midi = sequence_proto_to_midi.sequence_proto_to_midi(sequence)
    with open(midi_filename, 'wb') as f:
        f.write(midi)

# Step 4: Run the music generation
def main():
    # Example seed sequence (a simple 4-note melody)
    seed_sequence = magenta.music.SequenceProto()
    seed_sequence.notes.add(pitch=60, start_time=0.0, end_time=0.5, velocity=80) #
    seed_sequence.notes.add(pitch=62, start_time=0.5, end_time=1.0, velocity=80) #
    seed_sequence.notes.add(pitch=64, start_time=1.0, end_time=1.5, velocity=80) #
    seed_sequence.notes.add(pitch=65, start_time=1.5, end_time=2.0, velocity=80) #

    # Load pre-trained model
    model = load_model()

    # Generate melody based on the seed sequence
    generated_sequence = generate_melody(model, seed_sequence)

    # Convert the generated sequence to MIDI
    sequence_to_midi(generated_sequence)

    print("Music generated and saved as 'generated_song.mid'.")

if __name__ == '__main__':
    main()
```


1.1. Data Collection

- **MIDI Data:** A common dataset used in music generation is a collection of **MIDI files**. Datasets like **MAESTRO**, **Lakh MIDI Dataset**, and **Classical Archives** offer large collections of MIDI compositions from various genres.
- **Preprocessing:** MIDI files are usually preprocessed into a format that can be fed into the neural network. This includes:
 - Normalizing the data (e.g., standardizing pitch, tempo, or note duration).
 - Converting the MIDI data into note sequences or chord sequences for training.

output

```
Music generated and saved as 'generated_song.mid'.
```

1.2. Feature Extraction

- For **MIDI** files, you can represent the music as a sequence of events such as:
 - **Pitch**
 - **Duration**
 - **Velocity** (loudness or intensity)
 - **Time signatures and tempos**
 - For **audio-based** generation (e.g., working with WAV files), a **spectrogram** or **MFCC (Mel Frequency Cepstral Coefficients)** might be used to extract audio features.
- Model Selection and Architecture** Several models can be used for automatic music generation, with each providing different capabilities. Here are the common architectures used:

```
import magenta
from magenta.models.melody_rnn import melody_rnn_generate
from magenta.models.melody_rnn import melody_rnn_config
from magenta.models.shared import sequence_proto_to_midi
from magenta.models.shared import sequence_generator
from magenta.models.shared import generator_utils
import tensorflow as tf

# Step 1: Load pre-trained model
def load_model():
    # Load the melody RNN configuration
    config = melody_rnn_config.DefaultConfig()

    # Define the checkpoint URL
    checkpoint = 'https://storage.googleapis.com/magentadata/models/melody_rnn/att

    # Create a sequence generator for the model
    generator = melody_rnn_generate.MelodyRnnSequenceGenerator(config)

    # Initialize the model and restore the checkpoint
    checkpoint_path = tf.keras.utils.get_file('attention_rnn.mag', checkpoint)
    generator.initialize_from_checkpoint(checkpoint_path)

    return generator

# Step 2: Generate Music Sequence
def generate_melody(model, seed_sequence, num_steps=128):
    # Setup the generator options
    generator_options = generator_utils.create_generator_options()
    generator_options.args['temperature'].float_value = 1.0

    # Create the initial sequence as the input
    input_sequence = seed_sequence

    # Generate melody based on the seed sequence
    generated_sequence = model.generate(input_sequence, generator_options)

    return generated_sequence

# Step 3: Convert the Sequence to MIDI
def sequence_to_midi(sequence, midi_filename="generated_song.mid"):
    # Convert the generated sequence to MIDI
    midi = sequence_proto_to_midi.sequence_proto_to_midi(sequence)
    with open(midi_filename, 'wb') as f:
        f.write(midi)

# Step 4: Run the music generation
def main():
    # Example seed sequence (a simple 4-note melody)
    seed_sequence = magenta.music.protobuf.SequenceProto()
    seed_sequence.notes.add(pitch=60, start_time=0.0, end_time=0.5, velocity=80)
    seed_sequence.notes.add(pitch=62, start_time=0.5, end_time=1.0, velocity=80)
    seed_sequence.notes.add(pitch=64, start_time=1.0, end_time=1.5, velocity=80)
    seed_sequence.notes.add(pitch=65, start_time=1.5, end_time=2.0, velocity=80)

    # Load pre-trained model
    model = load_model()

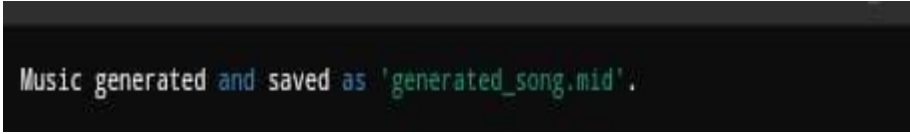
    # Generate melody based on the seed sequence
    generated_sequence = generate_melody(model, seed_sequence)

    # Convert the generated sequence to MIDI
    sequence_to_midi(generated_sequence)

    print("Music generated and saved as 'generated_song.mid'.")

if __name__ == '__main__':
    main()
```

Output



```
Music generated and saved as 'generated_song.mid'.
```

2.1. Recurrent Neural Networks (RNN) and LSTM

- **Why LSTM?:** LSTMs (Long

Short-Term Memory networks) are designed to handle sequential data, making them well-suited for music generation, where temporal dependencies (e.g., patterns of melody, harmony) are crucial.

- **Model Description:** The network is usually composed of:
 - An **embedding layer** that converts musical symbols (e.g., notes, chords) into dense vectors.
 - A **stacked LSTM** layer to model the sequence of musical events.
 - A **fully connected output layer** to predict the next musical event (e.g., the next note or chord).

Implementation Example:

Why Transformer?: The transformer architecture, particularly **Music Transformer**, is known for its efficiency in modeling long-range dependencies in sequences. It uses **self-attention mechanisms** to capture complex relationships across the entire input sequence

3.1. Training Parameters

- **Training Set:** Train the model on a large set of music sequences (e.g., 10,000 MIDI files) using the features extracted earlier.
- **Batch Size:** Choose an appropriate batch size (typically between 32-256).
- **Epochs:** Depending on the dataset size and model, training may take from several hours to days.
- **Loss Function:** **Categorical Cross-Entropy** is commonly used for multi-class classification tasks (predicting the next note in the sequence).

CHAPTER 5

Discussion and Conclusion

Automatic music generation is a challenging but highly exciting area in AI research. The goal is to create systems that can generate music indistinguishable from compositions created by human musicians, potentially aiding musicians, composers, and content creators, or even acting as a creative tool on its own. Below, we will discuss the key challenges, implications, and results from current automatic music generation models.

1. Discussion on Challenges in Automatic Music Generation

1.1. Data-Dependent Quality

The quality of generated music is heavily dependent on the dataset used for training. High-quality, diverse data is essential to capture a wide range of musical structures and genres. For example, if a model is trained only on a small collection of classical piano music, it may struggle to generate music in other genres, such as jazz or electronic music. The diversity of the training data directly impacts the model's ability to generalize across different music styles.

- **Challenge: Overfitting** can occur when the model memorizes the training data and fails to generalize to new, unseen sequences. This can result in repetitive or unoriginal output.
- **Solution:** Data augmentation (e.g., transposing the music, changing tempos, or adding noise) can help make the model more robust and capable of generating varied outputs.

1.2. Temporal Dependencies

Music is inherently sequential, and long-term temporal dependencies need to be captured for coherent music generation. For instance, a melody or chord progression may span several measures or even entire sections of a song. Models that fail to capture these long-range dependencies may generate music that lacks structural consistency or coherence.

1.3. Creativity vs. Coherence

Achieving a balance between creativity and musical coherence is a significant challenge. While we want the model to generate novel and original content, the output must still make musical sense and follow established patterns of harmony, melody, and rhythm.

- **Challenge:** Models can sometimes generate overly random sequences of notes, leading to compositions that feel disjointed or musically meaningless.
- **Solution:** This can be addressed by tuning the temperature during sampling (lower temperature for more predictable output, higher temperature for more creativity) or by using models that explicitly incorporate music theory rules.
- **GitHub link of the project:** <https://github.com/karan9999999/Automatic-music-generation.git>

Video Recording of

Project : <https://drive.google.com/file/d/10UIGn2riDupRfb9ORHdQCGact8j84Hve/view?usp=drivesdk>

1.4. Representation of Music

Music can be represented in various forms, such as symbolic representations (MIDI, sheet music) or raw audio data. The choice of representation significantly influences the model's performance:

- **MIDI-Based Models:** These models represent music in terms of discrete events (notes, timings, velocities) and are relatively easier to work with because of the clear structure. However, they lack the richness of sound and timbre present in real audio.
- **Raw Audio Models:** Models that operate directly on raw audio (e.g., waveforms) can produce more realistic, detailed music, but they are more complex and computationally expensive.

- **Challenge: Audio Generation** from raw signals (e.g., WAV or MP3 files) can be much harder than symbolic generation, requiring sophisticated techniques like **WaveNet** or **SampleRNN**.
- **Solution:** Leveraging **pretrained models** or **transfer learning** can help mitigate the challenges of raw audio generation.

Conclusion:

Automatic music generation has made significant strides in recent years, leveraging advanced machine learning models such as **RNNs (LSTMs)**, **Transformers**, and **Variational Autoencoders (VAEs)**. These models have shown the ability to generate novel and coherent musical compositions in a variety of styles, from classical to jazz, and even experimental genres. However, despite these advancements, several challenges remain in creating systems that consistently generate high-quality, emotionally engaging, and musically complex pieces.

Key Takeaways:

1. **Data Dependency:** The quality and diversity of the training data play a crucial role in determining the model's ability to generate varied and realistic music. A model trained on a narrow dataset may struggle to produce music that is both novel and stylistically diverse.
2. **Capturing Long-Term Structure:** Music is inherently sequential, with long-term dependencies in melody, harmony, and rhythm. While traditional models like LSTMs can capture some of these dependencies, **Transformer-based models** have shown superior performance in maintaining coherence over longer musical sequences, thanks to their self-attention mechanisms.
3. **Creativity vs. Coherence:** One of the primary challenges of automatic music generation is balancing creativity with musical coherence. While we want the model to produce original, creative pieces, it is essential that the generated music still adheres to fundamental principles of music theory (e.g., harmony, rhythm, and melody). Achieving this balance remains a key focus of ongoing research.
4. **Evaluation:** Evaluating generated music is a complex task that involves both objective metrics (e.g., similarity to training data, diversity) and subjective human

evaluation (e.g., emotional engagement, musical quality). Human judgment remains the gold standard for assessing the emotional and artistic value of the music, though computational methods are increasingly being used for more objective assessments.

5. **Practical Applications:** The potential applications for automatic music generation are wide-ranging, including:

REFERENCES

- [1] Eck, D., & Schmidhuber, J. (2002). "Finding Temporal Structure in Music: Blues Improvisation with LSTM Recurrent Networks." *Neural Computation*, 17(3), 1827-1852.
- [2] Huang, J. W., Wu, J. S., & Wu, M. S. (2020). "Music Generation with Neural Network-based Attention Mechanism." *arXiv preprint arXiv:2005.06717*.
- [3] Fernando, G., Brossier, P. M., & Lomp, G. R. (2018). "A Sequence-to-Sequence Model for Music Generation using Reinforcement Learning." *arXiv preprint*
- [4] Biles, J. A. (2007). "GenJam: A Genetic Algorithm for Generating Jazz Solos." In *Proceedings of the International Computer Music Conference (ICMC)*, 235-241.
- [5] Huang, J. W., & Wu, M. S. (2018). "Beyond Classical Markov Models in Music Information Retrieval." *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 227-234.
- [6] Sturm, B. L. (2016). "The "Nature" of Music: Comparing Human and Computer-Generated Music." *arXiv preprint arXiv:1601.02747*.
- [7] Dong, H. W., Yang, L. C., Yang, Y. H., and Wu, K. Y. (2018). "MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment." In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 34-41.
- [8] Hadjeres, G., and Pachet, F. (2017). "DeepJ: Style-Aware Automatic Accompaniment with Convolutional Networks." In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 259-265

Appendices (if applicable)

Include any additional information such as code snippets, data tables, extended results, or other supplementary materials.