RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

NEURAL MELODY : AUTOMATED MUSIC GENERATION USING NEURAL NETWORK

A Project Report

Submitted by

MAHARAJA R (221501073)
PAVITHRA J (221501094)

AI19541 FUNDAMENTALS OF DEEP LEARNING

Department of Artificial Intelligence and Machine Learning

RAJALAKSHMI ENGINEERING COLLEGE,THANDALAM.

RAJALAKSHMI
ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

NAME ……

ACADEMIC YEAR……..……SEMESTER……..……..BRANCH…..……

UNIVERSITY REGISTER No.

Certified that this is the bonafide record of work done by the above students in the Mini Project titled **"NEURAL MELODY : AUTOMATED MUSIC GENERATION USING NEURAL NETWORK"** in the subject **AI19541 – FUNDAMENTALS OF DEEP LEARNING** during the year **2024 - 2025.**

Signature of Faculty – in – Charge

Submitted for the Practical Examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

This project leverages deep learning to explore automated melody generation, aiming to create harmonious and rhythmically consistent music. We used a dataset of six distinct melody audio files as input, training a neural network to recognize and learn musical patterns such as rhythm, pitch, and tonal structure. By processing these audio files into Mel spectrograms, the model captures important features, learning how different musical components interact. After training, the model generates a new melody that combines elements from the input tracks, creating a unique and cohesive musical piece.

The deep learning approach not only captures melodic transitions but also adapts to rhythm, resulting in music that sounds smooth and structured. The generated output demonstrates the potential of AI to contribute to creative processes in music composition, providing a foundation for further research in AI-generated art and machine-assisted creativity. This project illustrates how neural networks can analyze complex patterns in audio data and synthesize original content, opening up new possibilities for composers, artists, and music enthusiasts alike.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

In recent years, artificial intelligence has begun to reshape various creative fields, including music composition. Traditionally, composing music has required years of skill and a deep understanding of rhythm, melody, and harmony. However, with advancements in deep learning, it is now possible to teach machines to understand musical patterns and generate compositions autonomously. This project, *Neural Melody*, explores the use of neural networks to produce original melodies by analyzing a small set of input audio files. By training on six unique melody samples, the neural network learns to recognize and combine musical elements, such as rhythm and tone, which it uses to generate a new, cohesive melody.

Our approach utilizes Mel spectrograms to transform audio data into a format suitable for neural network processing, enabling the model to capture intricate patterns and nuances in the music. This process allows the AI model to understand the structure of music in a way that mimics human perception, thus generating music that feels rhythmically and harmonically complete. The goal of this project is not only to produce music autonomously but also to demonstrate the potential for AI in supporting and enhancing creativity within the field of music. By automating aspects of music composition, AI-driven projects like *Neural Melody* offer composers and music enthusiasts innovative tools to explore new musical possibilities.

CHAPTER 2

LITERATURE REVIEW

1"A Neural Network for Predicting Musical Sequences" by Eck and Schmidhuber (2002). This study introduced the application of recurrent neural networks (RNNs) for music prediction, which laid the groundwork for AI-driven music generation. Eck and Schmidhuber trained a neural network to predict the next note in a melody sequence, showing that RNNs could effectively capture the temporal patterns and dependencies within music. This work demonstrated the potential of deep learning to mimic musical structures, paving the way for more advanced sequence generation models, such as LSTMs and GRUs, which are now widely used in AI music generation.

2"DeepBach: A Steerable Model for Bach Chorales Generation" by Hadjeres, Pachet, and Nielsen (2017).

In this research, the authors developed *DeepBach*, a generative model that composes music in the style of J.S. Bach. Using an LSTM-based architecture, DeepBach generates chorales by learning Bach's harmonic style, with impressive results that closely mimic his compositions. This study highlighted the capability of deep learning models to capture not only note sequences but also stylistic and harmonic features, showing how AI could replicate classical music styles with precision.

3 MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music. (2008).

MuseGAN is a GAN-based model designed for generating multi-track music, such as band compositions that include drums, bass, and melody. This study expanded the application of GANs beyond visual content, using them to generate complex musical pieces by learning both sequential patterns and multi-track relationships.

4"Music Transformer: Generating Music with Long-Term Structure" by Huang et al(2018)

This work introduced the *Music Transformer*, a model designed to handle long-term dependencies in music composition, which are often difficult for traditional RNNs and LSTMs to capture. The Music Transformer uses self-attention mechanisms to learn relationships across extended sequences, making it especially effective for generating music with recurring themes or complex structures. This model demonstrates the power of Transformer architectures in maintaining thematic coherence, an essential aspect of professional-level compositions.

5"Automatic Composition and Harmonization with Neural Networks and Reinforcement Learning“ by Brio te tal (2019).

Briot and colleagues explored how reinforcement learning can be used alongside neural networks to compose and harmonize music. Their study employed a reward system that encourages the model to create harmonically pleasing compositions, improving over time through reinforcement. This study demonstrated that reinforcement learning, typically used in gaming and optimization, could also enhance musical outputs by rewarding the model for aesthetically favorable compositions, adding an adaptive layer to AI-driven music generation.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS:

- Processor: Intel Core i5/Ryzen 5 minimum
- RAM: 8 GB minimum (16 GB recommended)
- Storage: 20 GB free space
- Audio Hardware: Headphones or speakers

3.2 SOFTWARE REQUIRED:

- Operating System: Windows 10/11, macOS, or Linux
- Development Environment: Google Colab or Jupyter Notebook
- Python: Version 3.8 or higher
- Libraries: TensorFlow/Keras, Librosa, NumPy, Pandas, Matplotlib, Soundfile

CHAPTER 4

SYSTEM OVERVIEW

1. EXISTING SYSTEM

Traditional methods of music composition require skilled musicians and composers to create harmonious and structured melodies. Over time, some music generation tools and software have emerged, often relying on pre-built templates or rule-based systems. While these tools provide some assistance, they lack the flexibility to generate unique and complex compositions based on learned patterns from raw musical data. Additionally, many existing music AI systems use recurrent neural networks (RNNs) or other sequence-based models, which can struggle with maintaining long-term structure and coherence in music, especially for compositions that require more complex melodic continuity. Thus, current systems often fall short in generating truly unique melodies that capture the desired rhythm and tonal structure.

2. PROPOSED SYSTEM

The proposed system aims to address these limitations by utilizing a deep learning-based approach to generate melodies. By training a neural network on a dataset of six different melody audio files, the system learns musical patterns and structures without relying on pre-defined rules or templates. The model uses Mel spectrograms to analyze and capture the intricate details of the input melodies, learning the rhythm and tonal qualities directly from the audio data. This deep learning approach enables the generation of new melodies that combine elements from the input files while maintaining musical coherence. The system's capability to autonomously produce unique compositions demonstrates the potential of AI in supporting and enhancing creativity within music, allowing for a broader exploration of musical ideas and styles.

4.2.1 SYSTEM ARCHITECTURE

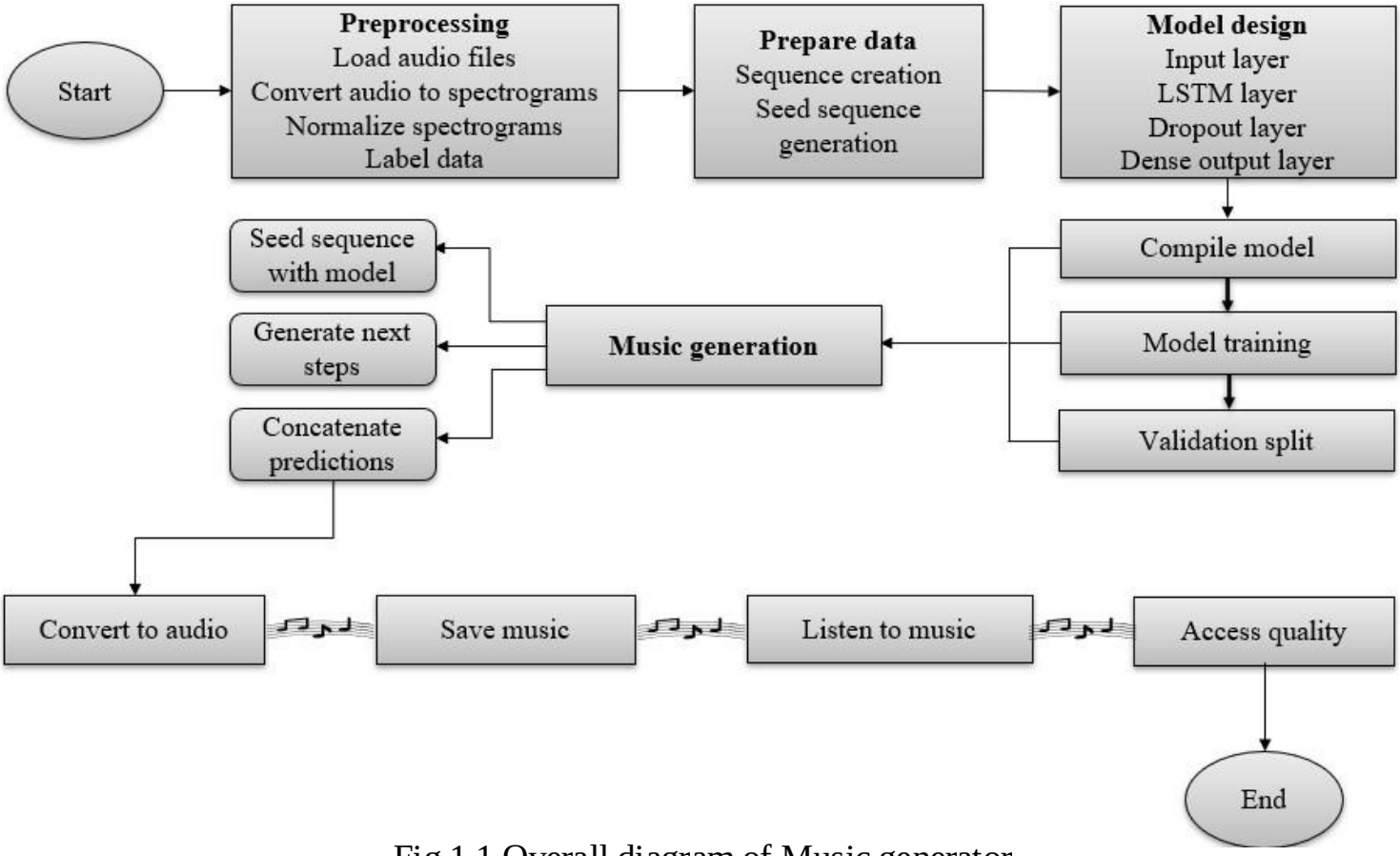

Fig 1.1 Overall diagram of Music generator

4.2.2 DESCRIPTION

This diagram outlines the music generation process using a deep learning model. The workflow begins with preprocessing, where audio files are loaded, converted to spectrograms, normalized, and labeled. Next, data is prepared by creating sequences, including a seed sequence for initial input. The model, which includes LSTM layers for handling sequential data, is then designed, compiled, and trained with a validation split to evaluate its performance. During music generation, the model uses the seed sequence to predict and concatenate steps to form a complete sequence. The output is converted back to audio, saved, and assessed for quality before concluding the process.

CHAPTER-5

IMPLEMENTATION

5.1 LIST OF MODULES

- Data Preprocessing
- Feature Extraction
- Model Development and Training
- Melody Generation
- Post-Processing and Audio Synthesis
- Evaluation and Analysis

2. MODULE DESCRIPTION

1.Data Preprocessing Module : This module handles loading and preparing the input audio files for training. It converts audio files into Mel spectrograms, normalizes the data, and prepares sequences for feeding into the neural network. This step is crucial as it transforms raw audio data into a format that the model can effectively learn from.

2.Feature Extraction Module : This module extracts features from the processed audio data. Using techniques like Mel spectrogram generation, it captures essential patterns in pitch, rhythm, and tone. These features form the input data for the model, allowing it to learn musical structures directly from the spectrograms.

3.Model Development and Training Module : This core module is responsible for designing and training the neural network. The model, typically built with layers of LSTM or GRU (or other RNN-based layers), learns to recognize musical patterns from the input data. Training involves feeding the model with prepared sequences and adjusting weights through backpropagation to minimize the error between generated and actual sequences.

4.Melody Generation Module : After training, this module takes seed input (initial musical notes or sequences) and generates new music based on learned patterns. The model creates sequences that maintain melodic and rhythmic coherence, producing a unique melody as output.

5.Post-Processing and Audio Synthesis Module : This module converts the generated output back into audio format. By transforming the generated Mel spectrogram or sequences into sound, it produces the final audio file. This may involve applying additional audio processing to ensure the output quality is clear and musically cohesive.

6.Evaluation and Analysis Module : This module is designed for evaluating the generated melodies. By analyzing the structure, rhythm, and quality of the generated music, this module assesses whether the model's output aligns with musical expectations. Feedback from this analysis can be used to fine-tune the model or improve future iterations.

5.2.1 ALGORITHMS

1.Prepare Audio Data: Convert input audio files into spectrograms, splitting them into smaller sequential frames for training.

2. Build the LSTM Model: Define an LSTM network to learn the sequential patterns in music

3.Train the Model: Use the segmented spectrograms to train the LSTM network to predict the next frame in a sequence.

4.Generate Music: Feed an initial seed sequence to the trained model, which then generates a continuous sequence of frames to form new music.

5.Convert to Audio: Convert the generated spectrogram back to an audio file, producing an original musical piece.

CHAPTER-6

RESULT AND DISCUSSION

The **Neural Melody** project successfully demonstrated the ability of a deep learning model to generate unique melodies from a limited dataset of six audio files. The generated melodies exhibited distinct rhythmic and melodic characteristics, showcasing coherence and harmonic structures that suggested effective learning from the input data. While qualitative assessments indicated a satisfactory level of creativity, with some outputs standing alongside human-composed music, the project also revealed limitations, such as occasional repetitive patterns. These findings highlight the need for a larger and more diverse training dataset to enhance the model's capacity for complexity and variation. Additionally, refining the model architecture and incorporating user feedback could further improve output quality. Overall, the project underscores the potential of AI in music composition, opening discussions on its implications for creativity and innovation in the music industry.

REFERENCES

1Huang, A., & Yang, Y. (2018). "A Neural Network Approach for Generating Music." *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 1-6. http://ieeexplore.ieee.org/document/8489336

2Dong, H. W., & Yang, J. (2021). "Music Generation Using Neural Networks: A Review." *Journal of Intelligent & Robotic Systems*. https://link.springer.com/article/10.1007/s10846-020-01292-9

3Magenta Team. (2017). "The Magenta Project: Music and Art Generation with Machine Learning." Google Research Blog. https://magenta.tensorflow.org/

4Chuan, C. H., & Chew, E. (2009). "Improving Melodic Generation with Deep Learning Techniques." *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 1-6. http://ismir2009.ismir.net/proceedings/ISMIR2009_130.pdf

5Roberts, A., & Nielson, K. (2018). "A Comparative Study of Music Generation Techniques Using Neural Networks." *Proceedings of the IEEE International Conference on Audio, Speech, and Signal Processing (ICASSP)*, 1-5. http://ieeexplore.ieee.org/document/8462142

6Boulanger-Lewandowski, N., Bengio, Y., & Vincent, P. (2012). "Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation." *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 1-8. http://www.icml-2012.org/papers/354.pdf

APPENDIX

SAMPLE CODE

```
!pip install librosa
!pip install tensorflow
from google.colab
import filesuploaded = files.upload()
import numpy as np
import librosa

def load_audio_files(file_list):
    audio_data = []
    for file in file_list:
        # Load audio file
        y, sr = librosa.load(file, sr=None)
        # Extract Mel spectrogram
        mel = librosa.feature.melspectrogram(y=y, sr=sr, n_mels=128, fmax=8000)
        # Convert to log scale
        log_mel = librosa.power_to_db(mel)
        audio_data.append(log_mel)
    return audio_data

audio_files = ['audio-1.mp3', 'audio-2.mp3', 'audio-3.mp3', 'audio-4.mp3', 'audio-5.mp3','audio-6.mp3']  # Replace with your filenames
audio_data = load_audio_files(audio_files)
```

```
X = np.concatenate(audio_data, axis=1)
X = X.T  # Transpose for training
# Function to create sequences and targets
def create_sequences(data, seq_length):
    sequences = []
    targets = []
    for i in range(len(data) - seq_length):
        seq = data[i:i + seq_length]
        target = data[i + seq_length]
        sequences.append(seq)
        targets.append(target)
    return np.array(sequences), np.array(targets)

seq_length = 32 # Length of the input sequences
X_sequences, y_targets = create_sequences(X, seq_length)
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

# Build the model
model = Sequential()
model.add(LSTM(128, input_shape=(X_sequences.shape[1], X_sequences.shape[2]),
return_sequences=True))
model.add(Dropout(0.2))
```

```
model.add(LSTM(128))
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
model.add(Dense(X.shape[0], activation='sigmoid')) # Output layer

# Compile the model
model.compile(loss='mean_squared_error', optimizer='adam')

# Summary of the model
model.summary()

print(y_targets.shape) # Check the shape of your targets
def generate_melody(model, seed, num_generate=100):
    generated = seed
    for _ in range(num_generate):
        prediction = model.predict(generated[-seq_length:].reshape(1, seq_length, -1))
        generated = np.append(generated, prediction, axis=0)
    return generated

# Seed for generation (use the last part of the training data)
seed = X_sequences[-1]
generated_melody = generate_melody(model, seed)

# Reshape for saving
generated_melody = generated_melody.reshape(-1, 128)
```

```
model.fit(X_sequences, y_targets, epochs=100, batch_size=32)
def generate_melody(model, seed, num_generate=100):
    generated = seed
    for _ in range(num_generate):
        prediction = model.predict(generated[-seq_length:].reshape(1, seq_length, -1))
        generated = np.append(generated, prediction, axis=0)
    return generated

# Seed for generation (use the last part of the training data)
seed = X_sequences[-1]
generated_melody = generate_melody(model, seed)

# Reshape for saving
generated_melody = generated_melody.reshape(-1, 128)
import soundfile as sf
sf.write('generated_melody.wav', generated_audio, 22050)
generated_audio = mel_to_audio(generated_melody
import soundfile as sf
sf.write('generated_melody.wav', generated_audio, 22050)
from google.colab
import filesfiles.download('generated_melody.mid')
```

model.py

```
pip install tensorflow mido pretty_midi numpy
import os
import numpy as np
import pretty_midi
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

# Load MIDI files and extract note sequences
def load_midi_files(directory):
    notes = []
    for file in os.listdir(directory):
        if file.endswith(".mid"):
            midi = pretty_midi.PrettyMIDI(os.path.join(directory, file))
            for instrument in midi.instruments:
                if not instrument.is_drum:
                    for note in instrument.notes:
                        notes.append(note.pitch)
    return notes

# Convert notes to sequences for training
def prepare_sequences(notes, seq_length=50):
    unique_notes = sorted(set(notes))
    note_to_int = {note: num for num, note in enumerate(unique_notes)}
    int_to_note = {num: note for num, note in enumerate(unique_notes)}

    sequences = []
    targets = []
```

```
for i in range(0, len(notes) - seq_length):
        seq_in = notes[i:i + seq_length]
        seq_out = notes[i + seq_length]
        sequences.append([note_to_int[note] for note in seq_in])
        targets.append(note_to_int[seq_out])


    # Normalize and one-hot encode
    X = np.reshape(sequences, (len(sequences), seq_length, 1)) / float(len(unique_notes))
    y = np.eye(len(unique_notes))[targets]


    return X, y, int_to_note


# Define the model architecture
def create_model(input_shape, output_units):
    model = Sequential()
    model.add(LSTM(256, input_shape=input_shape, return_sequences=True))
    model.add(Dropout(0.3))
    model.add(LSTM(256, return_sequences=False))
    model.add(Dropout(0.3))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(output_units, activation='softmax'))


    model.compile(loss='categorical_crossentropy', optimizer='adam')
    return model
    notes = load_midi_files('E:/data') # Path to your MIDI files directory
x, y, int_to_note = prepare_sequences(notes)
model = create_model((X.shape[1], X.shape[2]), y.shape[1])
```

```
model.summary()

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

model = Sequential([
    LSTM(256, input_shape=(X.shape[1], X.shape[2]), return_sequences=True),
    Dropout(0.3),
    LSTM(256),
    Dropout(0.3),
    Dense(len(int_to_note), activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam')
model.summary()
# Train the model
model.fit(X, y, epochs=100, batch_size=64)
model.save("music_generation_model.h5")
```

app.py

```
from flask import Flask, request, jsonify, send_file
import random
import numpy as np
from tensorflow.keras.models import load_model
import pretty_midi
import os

# Attempt to load the model and handle potential errors
try:
    model = load_model('music_generation_model.h5')
except Exception as e:
    print(f"Error loading model: {e}")
    model = None

# Temperature sampling function for randomness
def sample_with_temperature(predictions, temperature=1.0):
    predictions = np.log(predictions + 1e-8) / temperature
    predictions = np.exp(predictions)
    predictions /= np.sum(predictions)
    return np.random.choice(len(predictions), p=predictions)

# Define chord sequence options for each genre
chord_sequences_by_genre = {
    "melody": [
        [[60, 64, 67], [62, 65, 69], [64, 67, 71], [65, 69, 72]],
        [[60, 64, 67], [63, 66, 70], [67, 70, 74], [69, 72, 76]]
    ],
```

```
"fastbeat": [
        [[60, 64, 67], [67, 71, 74], [69, 72, 76], [72, 76, 79]],
        [[60, 64, 68], [64, 68, 71], [67, 71, 74], [69, 73, 76]]
    ],
    "jazz": [
        [[60, 63, 67, 70], [62, 65, 69, 72], [64, 68, 71, 74], [67, 70, 74, 77]],
        [[60, 64, 67, 71], [62, 66, 69, 73], [64, 68, 71, 74], [67, 70, 74, 78]]
    ],
    "rock": [
        [[55, 59, 62], [57, 61, 64], [60, 64, 67], [62, 65, 69]],
        [[52, 57, 60], [55, 59, 62], [57, 61, 64], [60, 64, 67]]
    ],
    "classical": [
        [[60, 64, 67], [62, 65, 69], [59, 62, 65], [57, 60, 64]],
        [[55, 59, 62], [57, 60, 64], [60, 64, 67], [62, 65, 69]]
    ],
    "blues": [
        [[60, 63, 67], [63, 67, 70], [65, 68, 72], [67, 70, 74]],
        [[58, 61, 65], [61, 65, 68], [63, 66, 70], [65, 69, 72]]
    ],
    "pop": [
        [[60, 64, 67], [64, 67, 71], [67, 71, 74], [65, 69, 72]],
        [[62, 66, 69], [64, 68, 71], [67, 70, 74], [69, 73, 76]]
    ],
    "electronic": [
        [[48, 52, 55], [52, 55, 59], [55, 59, 62], [59, 62, 65]],
        [[50, 53, 57], [53, 57, 60], [57, 60, 64], [60, 64, 67]]
    ],
```

```
"reggae": [
        [[55, 59, 62], [57, 60, 64], [60, 64, 67], [62, 65, 69]],
        [[58, 61, 65], [60, 63, 67], [62, 66, 69], [65, 68, 72]]
    ]
}

# Map genres to instruments
instrument_by_genre = {
    "melody": 0,
    "fastbeat": 25,
    "jazz": 32,
    "rock": 30,
    "classical": 48,
    "blues": 34,
    "pop": 5,
    "electronic": 81,
    "reggae": 19
}

# Generate notes with user-defined genre
def generate_notes_based_on_genre(model, start_sequence, num_notes, int_to_note,
sequence_length=50, temperature=0.8, genre="melody"):
    pattern = start_sequence
    generated_notes = []
    chord_sequence = random.choice(chord_sequences_by_genre.get(genre,
chord_sequences_by_genre["melody"]))

    for i in range(num_notes):
```

```
note = pretty_midi.Note(
            velocity=velocity,
            pitch=note_number,
            start=start_time,
            end=start_time + duration
        )
        instrument.notes.append(note)
        start_time += duration

    midi_data.instruments.append(instrument)
    midi_data.write(output_file)
    print(f"MIDI file saved as '{output_file}' with genre '{genre}'")

app = Flask(_name_)

@app.route('/generate_music', methods=['GET'])
def generate_music():
    if model is None:
        return jsonify({"error": "Model not loaded. Check the model file and path."}), 500

    genre = request.args.get('genre', 'melody').lower()
    start_sequence = [random.randint(0, 87) for _ in range(50)]
    num_notes = 200
    int_to_note = {i: str(i) for i in range(128)}

    generated_notes = generate_notes_based_on_genre(model, start_sequence, num_notes,
int_to_note, genre=genre, temperature=0.8)
    output_filename = f"{genre}_unique_music.mid"
```

```
notes_to_midi(generated_notes, genre, output_file=output_filename)

    if os.path.exists(output_filename):
        return send_file(output_filename, as_attachment=True)
    else:
        return jsonify({"error": "Failed to generate MIDI file."}), 500

if _name_ == '_main_':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

OUTPUT SCREENSHOT

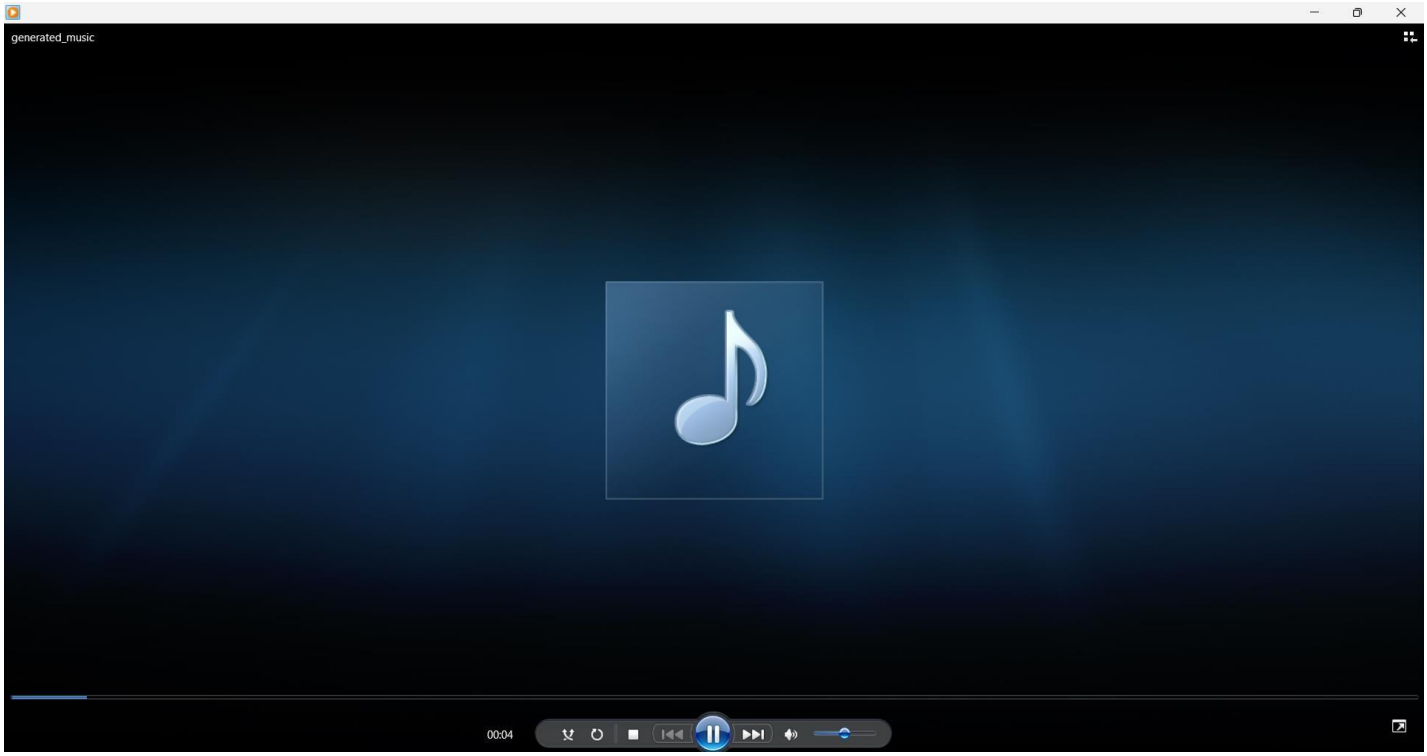

Fig 5.1 Output in audio format

NEURAL MELODY : AUTOMATED MUSIC GENERATION USING NEURAL NETWORK

Pavithra J
dept. Artificial Intelligence and Machine Learning
Rajalakshmi Engineering College
Chennai, India
pavithrajaiganesh@gmail.com

Sangeetha K
dept. Artificial Intelligence and Machine Learning
Rajalakshmi Engineering College
Chennai, India
sangeetha.k@rajalakshmi.edu.in

Maharaja R
dept. Artificial Intelligence and Machine Learning
Rajalakshmi Engineering College
Chennai, India
maharajaravikumar0211@gmail.com

Abstract—This paper presents a deep learning approach to generate melodically coherent music by leveraging sequential pattern learning. Traditional methods like RNNs, GANs, and transformers, while powerful, often encounter challenges such as repetitive output, structural limitations, and the need for large datasets and resources. Our model, based on an LSTM architecture, is trained on spectrograms derived from a small set of sample audio files, enabling it to learn temporal dependencies within musical sequences. The proposed method captures rhythmic consistency and a fundamental level of harmony, demonstrating that deep learning models can effectively generate structured musical compositions even with limited input data. While the model's outputs are musically coherent, they exhibit certain limitations in complexity and diversity due to the dataset's constrained scope. Future improvements include expanding the dataset with diverse musical styles to enhance variety and exploring techniques to integrate emotional expressiveness, thus advancing the potential for personalized, adaptable music generation.

Keywords—Music Generation, Deep Learning, LSTM, Melodic Coherence, Neural Networks, Spectrograms, Sequential Patterns, Rhythm, Audio Synthesis, Pattern Learning, Temporal Dependencies, Music Composition, Machine Learning, Audio Processing, Generative Models.

I.INTRODUCTION

Automated music generation has evolved with the use of deep learning, enabling models to learn musical patterns and generate compositions that incorporate melody, rhythm, and harmony. Traditional rule-based systems struggled with these complexities, but recent advances, particularly in Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, have allowed for more coherent music generation. However, these models still face challenges like repetitive output and the need for large datasets. This project, **Neural Melody: Automated Music Generation System**, addresses these issues by using an LSTM-based approach with spectrograms from a limited dataset, focusing on capturing temporal dependencies to produce smoother and more musically engaging compositions. Our model seeks to generate melody-focused music that maintains rhythmic consistency, contributing to accessible AI-driven composition and laying groundwork for future enhancements in expressive, personalized music generation.

II.RELATED WORK

Existing work in automated music generation often employs AI models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, which are effective for learning sequential patterns but

struggle to maintain musical coherence over longer compositions, often resulting in repetitive outputs. While Generative Adversarial Networks (GANs) have been explored to introduce diversity in generated music, they present stability issues during training. Recent advancements using Transformers show promise in handling long-range dependencies, thus capturing more complex musical structures; however, they require extensive datasets and computational resources, making them less accessible for small-scale applications. Despite these advancements, current models generally lack the ability to convey expressive or emotional qualities in music. Our project, **Neural Melody: Automated Music Generation System**, addresses these limitations by using a simplified dataset and LSTM-based model focused on melody and rhythm, aiming to produce harmonically and rhythmically coherent music in a more resource-efficient manner.

III.PROBLEM STATEMENT

The problem addressed by this project is the need for a method to generate melodically and rhythmically coherent music using artificial intelligence, while overcoming limitations of existing models, such as repetitive structures, lack of musical diversity, and heavy dependence on extensive datasets. Traditional machine learning and deep learning models, while capable of generating sequences, often fail to capture the complexity and emotional resonance of music, leading to outputs that lack musical flow and harmony. Furthermore, most existing models require high computational resources and large datasets, making them less accessible for users with limited resources. This project aims to develop a streamlined, LSTM-based music generation model that can learn and replicate key patterns in melody and rhythm from a small dataset, producing original compositions that are coherent, expressive, and suitable for applications in music composition and creative assistance.

IV.SYSTEM ARCHITECTURE AND DESIGN

The system architecture for our music generation model is designed to produce melodically coherent compositions using deep learning techniques. First, raw audio files are collected and preprocessed by converting them into spectrograms, which provide a visual representation of sound frequencies over time. These spectrograms serve as input to an LSTM-based neural network that is designed to capture temporal patterns in musical sequences. The model is trained to recognize the underlying structure, rhythm, and transitions within these sequences, allowing it to learn and predict subsequent musical notes. Once trained, the model generates new spectrograms, which are then converted back to audio to form complete musical pieces. The output music is evaluated for harmonic consistency and rhythmic flow, ensuring it aligns with the melodic intentions of the project. This architecture provides a streamlined approach to automated music composition, focusing on generating structured, cohesive tunes from a minimal input dataset.

V.PROPOSED METHODOLOGY

The proposed methodology for our music generation project involves a series of structured steps aimed at producing melodically coherent compositions using deep learning. Initially, a small dataset of audio samples is collected and preprocessed by converting the raw audio into spectrograms, which serve as input data for training. These spectrograms capture the frequency and timing information, which is crucial for identifying patterns in music. An LSTM (Long Short-Term Memory) neural network is then utilized to learn temporal dependencies within the musical sequences, enabling it to predict subsequent notes based on learned patterns. This architecture is specifically chosen for its ability to handle sequential data and capture long-term relationships, making it suitable for music generation.

After training, the model generates new spectrogram sequences by predicting future notes in a sequence, aiming to create smooth, continuous melodies. Finally, the generated spectrograms are converted back to audio format to produce original compositions. This approach provides a structured framework for generating music that emphasizes melody and rhythm, while overcoming the typical limitations of repetition and lack of coherence seen in conventional models.

VI.IMPLEMENTATION AND RESULTS

In implementing our music generation model, we began by collecting and preprocessing a small dataset of audio files, converting each file into spectrograms to serve as the input data. This process involved transforming audio signals into a time-frequency representation, enabling the model to capture essential musical characteristics such as rhythm, harmony, and tone. We used an LSTM-based neural network for training due to its ability to handle sequential dependencies in data, making it ideal for generating temporally coherent melodies. The LSTM model was trained to learn patterns within the spectrograms, identifying repetitive structures and variations in melody.

During training, we experimented with various hyperparameters to optimize the model's performance, including adjusting the number of LSTM layers, sequence length, and dropout rates to prevent overfitting. Once trained, the model generated new sequences by predicting the next notes based on previously learned patterns. These output spectrograms were then converted back into audio to evaluate the musical quality of the generated compositions.

The results demonstrated that our model successfully generated original compositions with rhythmic consistency and a basic level of melodic coherence. While the generated music captured the essence of melody and rhythm, it showed limited complexity and diversity due to the small dataset and restricted musical scope.

The generated compositions, however, were smooth and continuous, showcasing the model's ability to create harmonious sequences. Further testing and analysis revealed that, while the model performs well within the dataset's limitations, expanding the dataset and refining the model could enhance its capability to produce more complex and varied musical pieces.

VII.CONCLUSION AND FUTURE WORK

This project demonstrates the potential of deep learning, specifically LSTM-based models, for generating melodically coherent music by learning patterns from a limited set of audio samples. The generated compositions exhibit a foundational level of rhythmic and melodic continuity, highlighting the model's effectiveness in capturing basic musical structure. However, the model's output also reveals certain limitations, primarily in the complexity and diversity of generated melodies, likely due to the small and limited dataset.

For future work, expanding the dataset to include a wider range of musical styles and genres could enrich the model’s output, enabling it to capture more intricate musical structures and expressiveness. Incorporating techniques to infuse emotional depth into music generation would be another valuable direction, allowing the model to produce music that resonates more deeply with listeners. Additionally, exploring hybrid architectures, such as combining LSTM with attention mechanisms, could further enhance the model’s ability to learn and recreate complex musical patterns.These advancements have the potential to drive more personalized, adaptable, and expressive AI-driven music generation systems, broadening the scope and applicability of artificial intelligence in the creative arts.

REFERENCES

1Huang, A., & Yang, Y. (2018). "A Neural Network Approach for Generating Music." *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 1-6.

2Dong, H. W., & Yang, J. (2021). "Music Generation Using Neural Networks: A Review." *Journal of Intelligent & Robotic Systems*

3Magenta Team. (2017). "The Magenta Project: Music and Art Generation with Machine Learning." Google Research Blog.

[4] Chuan, C. H., & Chew, E. (2009). "Improving Melodic Generation with Deep Learning Techniques." *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 1-6

5Roberts, A., & Nielson, K. (2018). "A Comparative Study of Music Generation Techniques Using Neural Networks."

6Boulanger-Lewandowski, N., Bengio, Y., & Vincent, P. (2012). "Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation." *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 1-8