CS-351 Aritificial Intelligence

AI Music Generation with Genetic Algorithms and Neural Networks

Group Members :

- **Asad Ali 2022903**
- **Ammaid Saleem 2022344**
- **Hamayun Bajwa 2022189**
- **Mursaleen Khan 2022401**

Abstract

Artificial Intelligence (AI) has revolutionized numerous fields, and its application in music generation is no exception. This project report delves into the implementation of AI-driven music generation utilizing the MAESTRO dataset, TensorFlow, and other Python libraries. It outlines the methodology, tools, and results of exploring the dataset to establish a foundation for future work in AI-based music generation. Key aspects such as dataset preparation, library integration, and challenges encountered are discussed, culminating in a roadmap for developing advanced generative models.

Index

1. Introduction

1.1 Background

Music generation using artificial intelligence has become an exciting field of research, combining machine learning techniques with the art of music. The MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization) dataset provides a rich resource for training models on piano music. The dataset contains thousands of hours of aligned audio and MIDI files, making it ideal for machine learning applications in music.

1.2 Objective

The objective of this project is to load and preprocess the MAESTRO dataset, explore its contents, and set up a foundation for future music generation tasks using TensorFlow and Python libraries such as PrettyMIDI and FluidSynth.

---

2. Methodology

2.1 Tools and Libraries

The following tools and libraries are utilized in this project:

- **FluidSynth**: A software synthesizer for generating audio from MIDI files.
- **PrettyMIDI**: A library for parsing and analyzing MIDI files.
- **TensorFlow**: A deep learning framework for implementing and training models.
- **Seaborn and Matplotlib**: For data visualization.
- **NumPy**: For numerical operations.
- **Pandas**: For data handling and manipulation.

2.2 Dataset

The MAESTRO dataset (v2.0.0) is used in this project. It is downloaded from the Magenta dataset repository and contains both MIDI and corresponding audio files.

---

3. Implementation

3.1 Environment Setup

The project begins by installing the necessary libraries and tools:

```
!apt-get install fluidsynth  # Install the FluidSynth library
!pip install pyfluidsynth    # Python bindings for FluidSynth
!pip install pretty_midi     # For working with MIDI files
```

3.2 Library Imports

The essential Python libraries are imported to handle data manipulation, visualization, and model development:

```
import collections
import datetime
import fluidsynth
import glob
import numpy as np
import pathlib
import pandas as pd
import pretty_midi
import seaborn as sns
import tensorflow as tf
from IPython import display
from matplotlib import pyplot as plt
from typing import Optional
```

3.3 Seed Initialization

To ensure reproducibility of results, random seeds for TensorFlow and NumPy are set:

```
seed = 42
tf.random.set_seed(seed)
np.random.seed(seed)
```

3.4 Sampling Rate

The sampling rate for audio playback and generation is defined:
_SAMPLING_RATE = 16000

3.5 Dataset Preparation

The MAESTRO dataset is downloaded and extracted into a specified directory if it is not already present:

```
data_dir = pathlib.Path('data/maestro-v2.0.0')
if not data_dir.exists():
    tf.keras.utils.get_file(
        'maestro-v2.0.0-midi.zip',

origin='https://storage.googleapis.com/magentadata/datasets/maestro/v2.0.0/maestro-v2.0.0-midi.zip',
        extract=True,
        cache_dir='.', cache_subdir='data',
    )
```

---

4. Results and Analysis

4.1 Dataset Structure

Upon extraction, the MAESTRO dataset contains subdirectories organized by year, each containing MIDI and corresponding audio files. The dataset's size and content are ideal for machine learning applications.

4.2 Visualization

Further exploration and visualization of MIDI file contents using PrettyMIDI and Seaborn will provide insights into the distribution of note frequencies, durations, and other musical attributes. While this step is not shown in the provided code, it is an important next step for analyzing the dataset.

---

5. Discussion

5.1 Key Insights

- **MAESTRO Dataset**: Provides a comprehensive dataset for piano music, which is highly suitable for music generation tasks.
- **Tools Integration**: The integration of FluidSynth and PrettyMIDI enables seamless MIDI file analysis and audio generation.
- **Reproducibility**: The use of random seed initialization ensures consistent results across runs.

5.2 Challenges

- **Resource Requirements**: Processing and training on the MAESTRO dataset can be computationally intensive.
- **Audio Quality**: Synthesized audio quality depends on the sound font used in FluidSynth, which can be a limitation in some cases.

---

6. Conclusion

This project sets up a foundational pipeline for working with the MAESTRO dataset using Python libraries and tools. Future work will involve:

1. Preprocessing MIDI files for training a machine learning model.
2. Implementing a generative model (e.g., LSTM or Transformer) for music generation.
3. Evaluating generated music for creativity and quality.

By leveraging the MAESTRO dataset and advanced deep learning techniques, this project contributes to the field of AI-based music generation.

---

7. References

- MAESTRO Dataset: Magenta MAESTRO Dataset
- PrettyMIDI Documentation: PrettyMIDI
- FluidSynth Documentation: FluidSynth
- TensorFlow Documentation: TensorFlow