Assignment: Homework 7

How to Hand It In

1. Put all your solutions in one folder. Compress this folder and name it <firstname>_<lastname>_HW7.zip. For example, “Zhiyao_Duan_HW7.zip”.
2. Make sure you have a report to show graphs and explain your answers, in addition to source code.
3. Cite resources (e.g., external code repo) in your answers.
4. Submit to the corresponding entry on Blackboard.

When to Hand It In

It is due at 11:59 PM on the date specified on the course calendar. **Late assignments will receive a 20% deduction of the full grade each day.**

Background

Melody generation is a specialized topic within music generation. It aims to generate a monophonic sequence of notes with interesting melodic and rhythmic patterns to form a melody. This research has practical applications in music composition, music education, and entertainment. The methods for generating melodies vary widely, from systems based on predefined rules to sophisticated machine learning algorithms including recurrent neural networks (RNNs) and transformers. Specifically, Gated Recurrent Unit (GRU) models are favored for their efficiency in sequence-based tasks such as speech recognition and machine translation, thanks to their ability to handle long-term dependencies. Through this homework, you will acquire practical experience in utilizing GRU models for the purpose of melody generation.

Core Concepts of Recurrent Neural Networks (RNNs):

Processing Sequential Data: RNNs are specially designed for handling sequential data, incorporating a memory component that retains information from past inputs. This capability is crucial for capturing temporal dependencies, a fundamental aspect in music analysis and generation where the relation of a current musical note or chord to its preceding context is important.

Gated Recurrent Units (GRU): GRUs are one type of RNNs, developed to address the vanishing and exploding gradient problems common in standard RNNs. GRUs simplify the memory cell concept of LSTMs without compromising the ability to capture long-term dependencies. This makes GRUs adept at maintaining relevant information throughout extended data sequences, which is particularly beneficial for ensuring continuity in long musical compositions.

Encoding and Generating Data: In music generation, RNNs are fed with data such as notes, chords, or encoded musical structures. Through the learning process, the network identifies patterns and relationships within the data, enabling it to generate new musical sequences. In a causal generation paradigm, the generation process can begin with an initial sequence

input (i.e., seeding sequence) to the model, and then iteratively generates future musical elements, resulting in a novel composition.
Model Training Considerations: Effective training of an RNN for music generation involves meticulous adjustment of hyperparameters, such as sequence length, batch size, and learning rate. The training phase leverages a dataset of MIDI files or encoded musical information, teaching the model to recognize the complex patterns and structure characteristic of music.

Dataset: This dataset (attached dataset folder) contains 50 songs of MIDI format from popular singers like Ariana Grande, Chainsmokers, and The Weeknd.

Problems (10 points in total)

1. (1.5 points) Note Extraction

- Load the melody notes from each MIDI file and store them as a structured pandas DataFrame. Each row should represent a note with attributes pitch, step (i.e., time since the previous note's start time), and duration.
- **Hints**:
    - Use the pretty_midi.PrettyMIDI function to load a MIDI file.
    - Access the notes of the first instrument in the MIDI file. This is the melody line.
    - Iterate through the notes and calculate the 'step' by subtracting the start time of the current note from the start time of the previous note.
    - Store the attributes (pitch, step, duration) in lists and then convert these lists into a pandas DataFrame.

2. (1.5 points) Prepare Sequential Data for Training

- Write a function to transform the obtained DataFrame from the previous question into sequences and targets suitable for training an RNN. The sequences are sequences of notes in this assignment.
- **Hints**:
    - Decide on a fixed sequence length (e.g., 32 notes). This will be the number of notes in each input sequence to your model.
    - Create input sequences by taking consecutive notes of the chosen sequence length from the DataFrame.
    - The target for each sequence should be the note immediately following the sequence.
    - You may want to normalize the pitch values from [0, 127] to [0, 1].

3. (1.5 points) Design the RNN Model

- Design a GRU model that takes the sequences generated in Question 2 as input and predicts the next note.
- **Hints**:
    - Define the input shape of your model, based on the sequence length and the number of features per note.
    - Add GRU layers to your model. Experiment with adding multiple GRU layers and adjusting their number of units.
    - Use a Dense layer with a Softmax activation function to predict the next note's pitch distribution (i.e., a 128-d vector with each dimension corresponding to a MIDI number from 0 to 127). Add separate Dense layers for predicting the next note's step and duration (i.e., two real numbers).

4. (2 points) Train the Model

- Train the RNN model using the prepared dataset.
- **Hints**:
    - Split the dataset into train (1-40), validation (41-45) and test subsets (46-50).
    - Train the model with an appropriate optimizer (e.g., 'Adam') and loss function (e.g., cross entropy for pitch prediction, squared error loss for step and duration prediction).

5. (1 points) Generate Music with the Trained Model

- Use the trained model to generate a sequence of notes by iteratively predicting and appending the next note to the input sequence.
- **Hints**:
    - Start with a seed sequence of notes (i.e., the first 20 notes) from each of the test MIDI files, i.e., MIDI files No. 46-50.
    - Predict the next note using the model, append this note to the input sequence, and then feed it to the model to predict the following note. Repeat till the desired length of the melody. Note that the desired length can be much longer than the sequence length during training.
    - Listen to the generated melodies and compare with the ground-truth melodies. Are you satisfied with the generation? What aspects of the melodies would you like to improve?

6. (1 points) Convert Generated Note Sequences Back to MIDI

- Implement a function that takes the note sequences generated in Question 5 and converts them back into MIDI files.
- **Hints**:
    - Use the pretty_midi library to create a new MIDI file and add a single instrument to it.
    - Iterate over the generated notes, creating a pretty_midi.Note object for each note and adding it to the instrument's list of notes.
    - Use the pretty_midi.PrettyMIDI.write method to save the MIDI file. Make sure to upload your generated MIDI file during submission.

7. (1.5 points) Evaluation: Employ the following metrics to compare your generated melody with the ground truth: Pitch Class Histogram (PCH), Average Pitch Interval (PI), and Average Inter-Onset Interval (IOI). For each test piece, perform the following steps:

1. Calculate all metrics at the piece level for the generated melody of each test piece, and compare them with those calculated from the ground-truth pieces.
2. Calculate the PI and IOI metrics over time with a moving window. Describe changes of these metrics over time.

- These metrics are described in detail on page 5 of this paper: https://musicinformatics.gatech.edu/wp-content_nondefault/uploads/2018/11/postprint.pdf

- The metric calculation code is implemented in the following library: https://github.com/RichardYang40148/mgeval