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 report graphs and explain your answers, in addition to source code.
3. 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 subfield of music generation that focuses on the automatic generation of melodies using artificial intelligence techniques. The goal of melody generation is to create a sequence of musical notes that sound pleasant and (implicitly) satisfy melodic and rhythmic structures. It has applications in music composition, music education, and entertainment industries. Melody generation algorithms can take a variety of forms ranging from rule-based systems to advanced machine learning models such as recurrent neural networks (RNNs) and transformers. In particular, Long Short-Term Memory (LSTM) models are popular neural network architectures used in sequence modeling tasks like speech recognition and machine translation. In this homework, you will gain hands-on experience using LSTM models to generate melodies.

Dataset

For this assignment, you will be working with the IrishFolk dataset. The original dataset comprises 24,000 Irish folk melodies in MIDI format, out of which we have chosen 8,000 for this particular homework. Each song has been quantized to a 16th note grid and the data set is divided into three subsets:

- Training set: irish_train.pkl, with 7,000 songs
- Validation set: irish_train.pkl, with 1,000 songs
- Testing set: irish_test.pkl, with 3 songs with different lengths - a short song (4 measures), a medium-length song (32 measures), and a long song (115 measures). The diverse lengths of the testing data allow you to observe the model's performance across different generation ranges.

In the datasets, each song is represented as an array of integers (tokens) that correspond to MIDI numbers, with each value having a duration of a 16th note. The MIDI numbers range from 0 to 128, where values 0 to 127 represent note onsets with pitch value of the MIDI number, and 128 denotes a hold (i.e., continuing the previous note).

The dataset is stored using the pickle package, and you can load it with the following code:

```
import pickle

# Replace 'path_to_dataset' with the actual file path of the dataset
with open('path_to_dataset', 'rb') as file:
    data = pickle.load(file)
```

Problems (10 points in total)

1. (2 points) Implement a PyTorch **DataLoader** for the IrishFolk dataset. The objective of this DataLoader is to generate batches of training samples, where each batch comprises N input sequences and their corresponding targets. The DataLoader also requires a parameter, L_in, representing the input sequence length for the LSTM model. To create the DataLoader, follow these steps:
    a. Concatenate all the measures from all songs into a single long list.
    b. Truncate the list into non-overlapping segments of length L_in.
    c. Use these segments as input X and their subsequent token as the training targets y.
    d. Encode the MIDI values using one-hot encoding with a dictionary size of 48, as there are 48 distinct MIDI numbers in the dataset. Apply the following conversion table for this encoding:

MIDI Values	Type	One-hot Encoding
48	Pitch onset	[1, 0, 0, 0, …, 0, 0]
49	Pitch onset	[0, 1, 0, 0, …, 0, 0]
50	Pitch onset	[0, 0, 1, 0, …, 0, 0]
…	Pitch onsets	…
94	Pitch onset	[0, 0, 0, 0, …, 1, 0]
128	Hold	[0, 0, 0, 0, …, 0, 1]

48 Dimensions

2. (2 points) Implement a **unidirectional LSTM-based model** for generating melodies. The architecture of the model consists of a stack of two LSTM layers, each with 512 hidden units, followed by a fully connected layer with 512 units. To prevent overfitting, include a dropout layer with a dropout probability of 0.1. Finally, add another fully connected linear layer with softmax that outputs a 48-unit sequence representing the probability distribution over the possible next notes.

3. (2 points) Implement the **training loop**. Use the Adamax optimizer with a learning rate of 0.01. Train the model on the dataset for 100 epochs. Report prediction accuracy on the validation set. Plot “Training Loss vs Epoches” and “Accuracy vs Epoches”.
Note: torch.nn.CrossEntropyLoss is equivalent to the combination of LogSoftmax and NLLLoss.

4. (1.5 points) After training the generation model, implement the **inference code** to generate new melodies. During inference, the model takes a "seed sequence" as input and generates the 48-d encoding of the next note in the melody sequence, representing the probability distribution of the 48 tokens. Choose the note with the highest probability. We provide three "seed sequences" in the test dataset. For each test song, input the first L_in notes of that piece and generate new notes until reaching the length of the ground truth melody.
Note: You can play with generating melodies with an arbitrary length from the seed sequences, but make sure you report results using the ground-truth lengths.

5. (1 point) **Subjective Evaluation**. Once you have generated a sequence of notes using the trained LSTM model, you need to visualize it using the matplotlib library or other visualization tools of your choice. Next, implement a function to convert the generated sequences back to MIDI files using libraries like music21, pretty_midi, or mido. These libraries also offer functions for synthesizing the MIDI file into an audio waveform, enabling you to listen to the generated melody. Evaluate the generated sequences based on their musical quality and coherence by listening and looking at the visualization by yourself and describe your findings.
   Note: The key to the conversion from generated token sequences to MIDI is to "glue" tokens into notes, i.e., "hold" tokens should be treated as the continuation of the previous note.

6. (1.5 points) **Objective Evaluation.** Employ the following metrics to compare your generated melodies with the ground truths: Pitch Class Histogram (PCH), Average Pitch Interval (PI), Note Count (NC), and Average Inter-Onset Interval (IOI).
   For each test piece, perform the following steps:
   a. Plot the PCH for the entire piece and compare the PCH of the ground truth with that of the generated melody.
   b. For PI, NC, and IOI, compute each metric in every two measures for both the ground truth and the generated melody. Plot them as curves throughout each piece and compare the curves between the ground-truth and generated melodies.
   c. Compare the metrics across the three test songs and state your findings.

   Note: These metrics are described in detail on page 5 of this paper:
   https://musicinformatics.gatech.edu/wp-content_nondefault/uploads/2018/11/postprint.pdf
   Note: The metric calculation code is implemented in the following library:
   https://github.com/RichardYang40148/mgeval
   Note: The PI metric requires the python-midi package which cannot be installed in Python 3. You will need to implement this metric yourself based on the paper above.