AI Final Project

- ## Introduction

Recently, music has become one of the most popular topics that people mainly discuss. However, with this huge population, only a few are able to actually compose, which requires lots of experience and inspiration. What's more, for those who can't take advantage of some tools, transformation from written numbered musical notations to sheet music can also be complicated.

Therefore, in our project, we provide a generative AI that can generate the remaining numbered musical notations based on what the user provides, and help the user transform to sheet music if needed.

To ensure the performance of the model, we only focus on the piano part.

- ## Literature Review/Related work

We found a work using LSTM to do music generation. They use LSTM to generate piano pieces based on a song by Chopin.

The differences we made are mainly on input data and model structure. We use only notes without chords to prevent the size of the notation combination from being too large. Besides, we included multiple songs from different musicians to diversify. In model structure, we increase the number of layers and lessen the neurons in each layer.

- ## Dataset

A set of midi files are included in our project, which consists of more than 300 files. Besides loading data in midi files, we have to complete notes extraction with instrument selection as well as rare notes deletion. In addition, storing the data into csv can help reduce the time taken loading all the midi files.

- Baseline

We take the work from kaggle as the baseline, which generated music with a song from Chopin. After modifying the model and considering more songs from different composers, we expect a more diverse result.

- Main Approach

Data preprocess:

We use music21 tools to extract notes from midi files. Besides, to avoid some performance issues, we delete the notes that appear less than the average of appearance times.

Train:

The training process begins with command-line argument parsing to determine whether to read the notes from a csv file. If the argument (read_csv) is set to True, it reads the list of notes for each from the (preprocess) module. If it is false, we will load the midi file and train it from the beginning.

We use (n_vocab) to represent a unique count of notes which is determined. This is used to normalize the notes data later.

The function (prepare_sequences) is called with the note indices and n_vocab to prepare the training data. This function creates sequences of a certain length from the notes data. For each sequence of length (sequence_length) from a song, it appends the sequence(normalized by n_vocab) to the input data and the next note(also normalized) to the output data. Here we use one-hot entropy to append the sequence because it's hard to determine the relation when those musical notes become the number. For example, we transform ‘A’ to ‘1’ and ‘B’ to ‘2’ and so on. However, we can't say that ‘B’ is two times larger than ‘A’. That's the reason why we use one-hot entropy.

The function(create_network) is used to create the structure of the LSTM neural network model. The model consists of three LSTM layers. Every LSTM has a Dropout layer under it to reduce overfitting, a Dense layer to match the number of unique notes, and a softmax Activation layer. The model is compiled with a loss function(categorical_crossentropy) . This optimizer of our multi-class classification problem is ‘Adam’.

We also keep the model state in our training process. This can reduce our training time. That is, if there is a file named ‘model_weights.h5’ in the directory, the weights from this file are loaded into the model.

Finally, we try the different epoch numbers and loss value respectively. This is very hard because our dataset is very huge. We need to spend 1.5hr to train only one epoch.

● Evaluation Metric

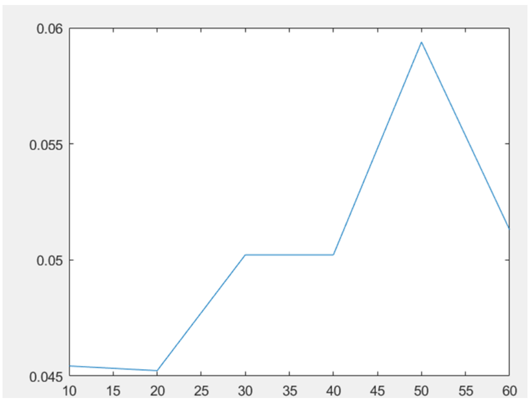
The model produces a new note based on five given notes. While evaluation, a random sample of a section of the input was considered as the ground truth to evaluate the performance. Even though the accuracy is 5.1%, it’s still much higher than randomly guessed, which is 1.2%, considering there are more than 80 kinds of notes. Besides, the diversity of generated notes is also our focused point. The diversity score(number of kinds of notes in output / number of kinds of notes in input) is 78.3%, which represents the flexibility of the model to produce different notes based on different input.

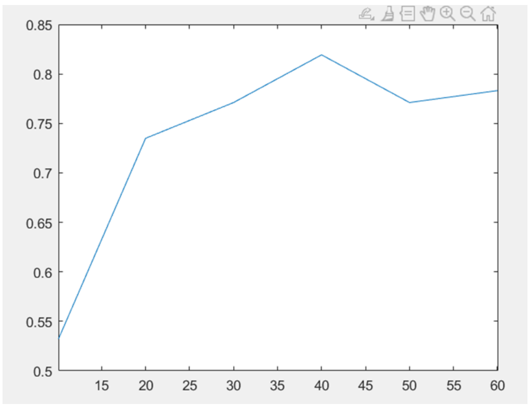
● Result & Analysis

Five randomly selected notes commence the generation with recursion to produce the remaining notes as well as build midi files. However, cycles emerge because the given five notes will likely produce the same notes. Therefore, we apply the judgment to prevent the production from cycles. If there is a cycle, three random notes are added to compel the model to output different results.

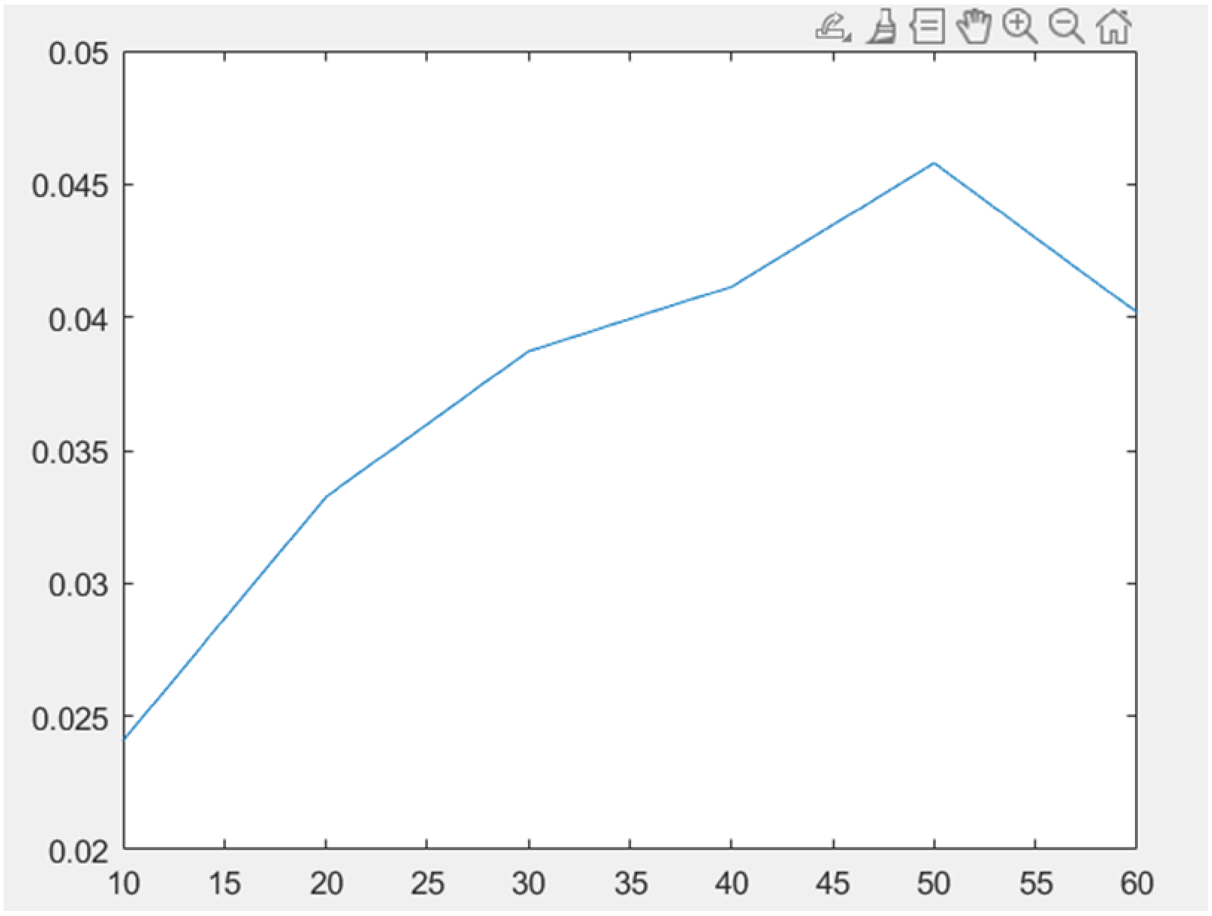
- ## Error Analysis

We have plotted the relationship between number of epochs and accuracy as well as note diversity.

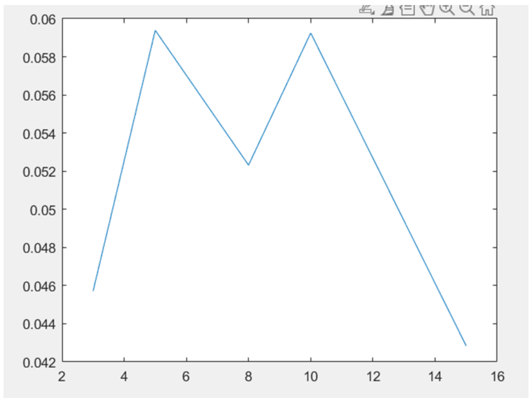


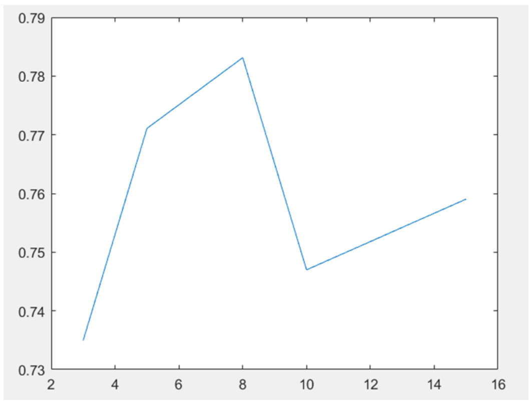


To combine these two figures, we discovered that the performance is better when epoch is 50.

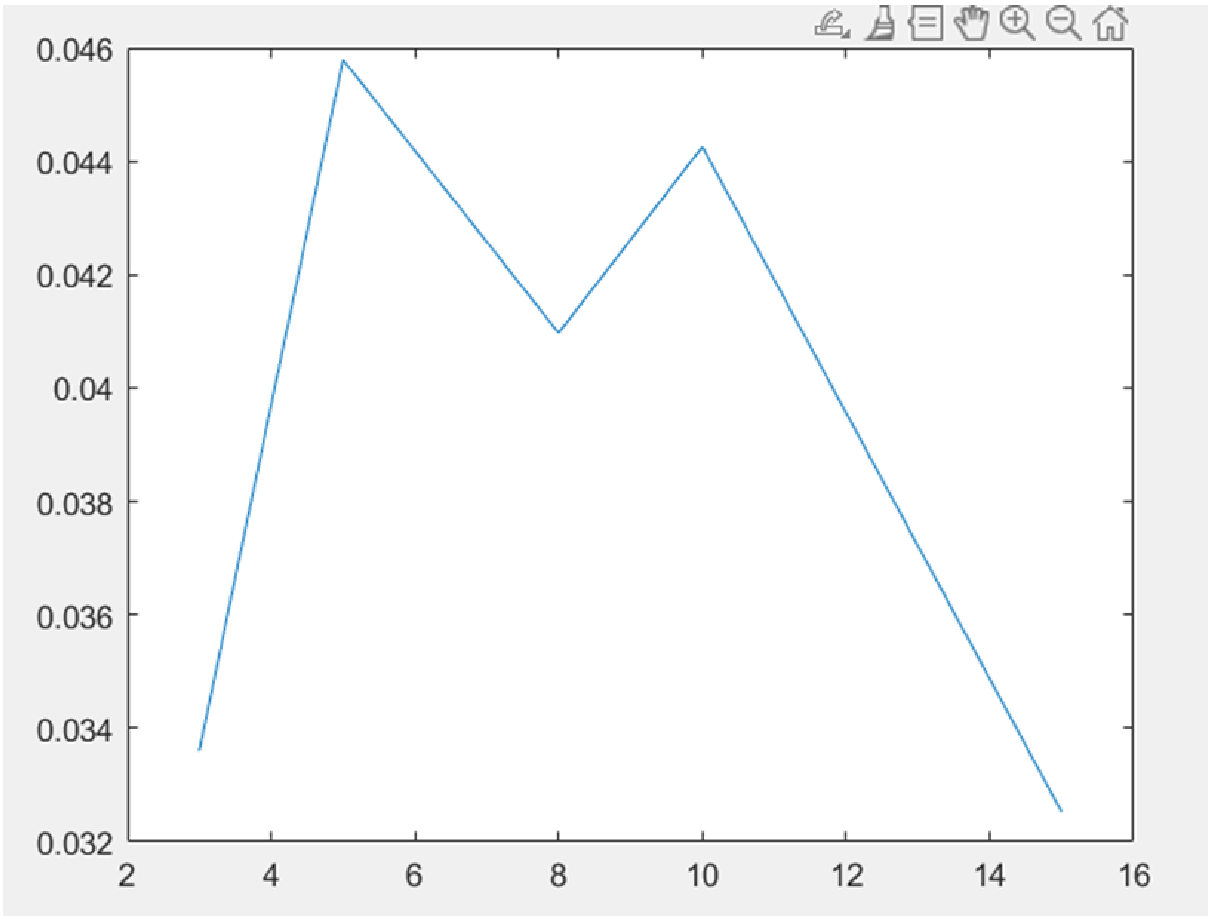


In addition, we also analyzed the same plots with sequence length, i.e. how many notes needed to generate a new note.





In observation, it turns out to be better when length equals to 5.



- Future Work

  In the future, we might be able to generate numbered musical notations of left hand and right hand independently to create more sound and diverse music. Additionally, we only include pitch in this project. There are some other data that can be considered, like volume or pitch space differences.

- Code

  While deleting rare notes, we use the average recurrence of each song as a threshold.
  There are two arguments with default can be set. First, you can read the csv file we provide to use those notes. Otherwise, loading your own midi files is feasible. Second, you can decide to train your own model with some modification in model.py. Otherwise, using our pretrained model is the default choice.

https://github.com/Mike1ife/AI-Final-Project

- Contribution of each member

戴廣逸: 25%
馬玉錚: 25%
陳昱佑: 20%
葉欲禾: 30%

- References

https://www.tensorflow.org/tutorials/audio/music_generation
https://github.com/salu133445/musegan
https://www.kaggle.com/code/karnikakapoor/music-generation-lstm/notebook