Appendix 2: Common FAQs

1. Which are some of the changes/challenges that existed in the original proposal? (tf? Format? etc)

    a. The author of this paper originally created this project in tensorflow 1. However, due to the compatibility, we eventually switched to a pytorch version from https://github.com/cjbayron/c-rnn-gan.pytorch.git
    b. In the original proposal we have identified that may occur during our implementation. However, we have met more serious problems relating to the music genre and data collected. (such as the data pre processing is not as ideal as the original data. Surprisingly, we the given data, the output is not ideal either. We have been working on our best to see if we could come out with a similar result as in the sample set given, however, upto now, the outcome is not as expected.
    c. A lot of details are not mentioned in the paper and it is hard to modify it.

2. What references (papers, libs, implementations) that you have used?
   We have used midi, pytorch….
    a. Pytorch ver. https://github.com/cjbayron/c-rnn-gan.pytorch.git
    b. FT ver. https://github.com/olofmogren/c-rnn-gan
    c. paper https://arxiv.org/abs/1611.09904
    d. Midi (python library)
    e. FL Studio 20 (DAW, for demonstration)
    f. Spire (VST, for demonstration)

3. What have you done in experimenting and modifying the original implementation? And why?
    a. We have explored its data pre-processing, c-rnn-gan structure
    b. To make them fit into our data set and training, we have changed all the training samples into our 204 samples of 8-bits music.
    c. To make it easier, we made them all in one genre and one composer
    d. There is a validation list and a test list. We have been modifying these two lists to meet different training expectations.
    e. To test them out, we have been trying on single game music set for example Zelda.
    f. We changed the resolution(mentioned in proposal) to lower value for a better tempo

4. What is the architecture of this model (visualization - layers, sequences, etc) - and how and why it is different from the original paper's architecture, if any?
    a. In paper: G: LSTM->Dense

b. in code: G: Dense->LSTM->dropout(0.6)->LSTM->Dense

c. In paper: D: LSTM->Dense
d. in code: D: dropout(0.6)->LSTM->Dense

5. How is this architecture different from other common music generation models like Char-RNN, VAE, etc? And what are its potential advantages/disadvantages?
    a. It is designed to be capable of dealing with complex music structures without overfitting and etc.

6. How do you preprocess the data? What features were extracted and fed into the neural network?
    a. To preprocess the code, we extracted out the note, tempo, time, and velocity.
    b. We will first normalize them into fitting format then compact them into an object
    c. In training phase, we will call this object to get the tuple that contains all the preprocessed information.

7. Why did the generated music not resemble the style of the original music? (reasons behind the high discriminator accuracy?) How you could improve it by changing how the data is fed or the architecture (due to time and knowledge limit, we may not have that time to improve it in this project)?
    a. The original music is often used as background music, we have tried our best to collect 200 music pieces that are in the similar style. However, this model still cannot handle that much complexity.
    b. If continuing this project, we would start with single track music pieces then add on complexity

8. Why did this model fail in producing melodic music? Is this based on the architecture of the model? If it is, how?

    This GAN model cannot transform chords + noise into new chords. But it's probably solvable, which needs efforts beyond the timeframe of this project. We'll try to continue improving this model (where the paper was published in 2016) to see if it can make even better music!