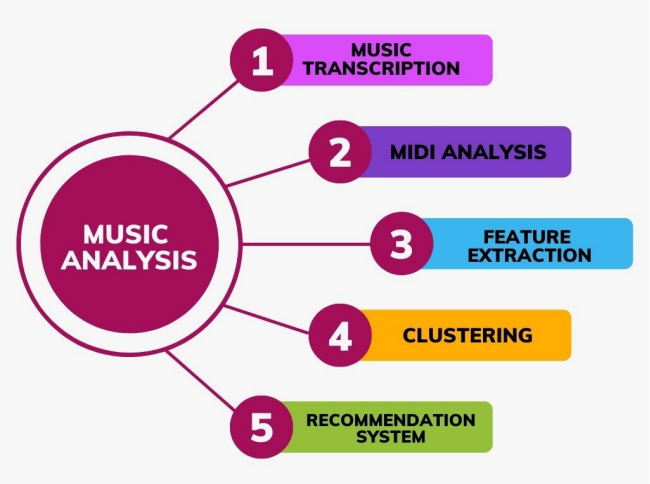
BrainStation Data Science

Music Transcription

Shi Yang Lee

Introduction

Music transcription is the process of notating a piece of music. The demand for music transcription arise from the the domain of music production and music streaming services. The purpose for this project is to explore the possibility of end-to-end music analysis, which includes music autotranscription as the first step.



The figure represents the grand idea of the project. The goal of music transcription includes converting raw audio file to computer recognizable MIDI (musical instrument digital interface.

Features such as notes distribution, keys, tempo and instruments can then be extracted by MIDI file analysis and feature extracion. These information can then be used to perform clustering, for example, clustering by sentiments or categories.

The final goal is then create an end-to-end pipeline to analyze raw music files, and producing recommendation system. This music analysis pipeline can be integrated with recommendation system based on customer's input (songs liked, skipped etc)

Background

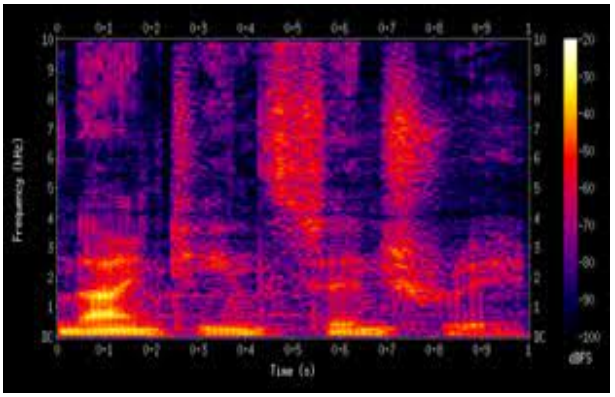
The dataset we will be working with is the OrchideaSOL and Musicnet. The OrchideaSOL dataset includes around 13,000 raw audio files, with orchestral instruments with wide range of notes. Each audio files contains sound with one instrument playing a single note for a short period of time.

The MusicNet dataset includes 330 classical music audio files, with exact notes labeled and instruments on the frames.

The reason this project was chosen as a machine learning project is the existence of neural network. There exists neural network architecture that are able to extract temporal and spatial information, and produce classification, specifically, we utilized CNN (convolutional neural network) and RNN (recurrence neural network) in this project. CNN have had huge suggest in handwriting recognition (MNST dataset), and recurrence network performs exceptionally in language translation, and dataset with sequential information.

Preprocessing

The first step we need to do before feeding data into the models is to convert our raw wav files into a spectrogram. Spectrograms are a dimensional matrix, which divided the audio by discrete timesteps and frequency bins. The color coding represents the magnitude.



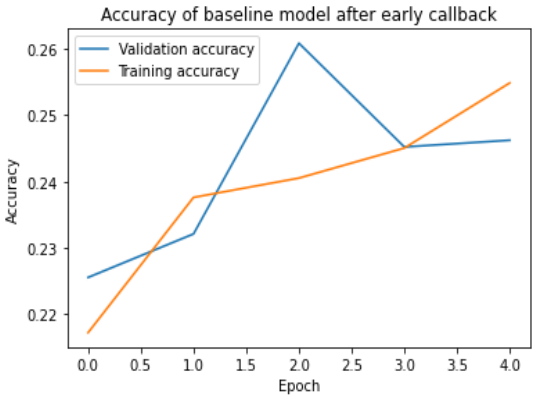
By converting our audio files into spectrograms, our models are able to pick up the magnitude relation across the frequency and time bins.

Since we are feeding out model into neural networks, it is essential for us to further augmentate our data. We will require our model to recognize the instrument under different environments and variations for the same instrument. We have implemented adding noise, audio masking and sihfting in our data preparation. See appendix for more information

Models

For the exact structure of the models, please refer to the appendix.

Baseline model



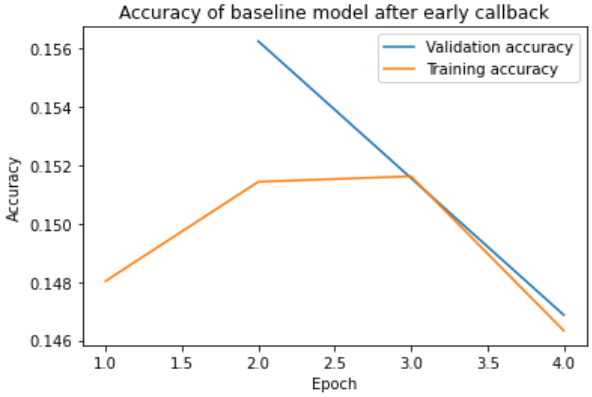
The first model we implemented is an baseline model of CNN. This model was trained on OrchideaSOL to classify different instrument in the audio files. The baseline model only served as a quick evaluation of how well CNN were performing in such problem.

The model consisted of a simple convolutional layer, extracting feature from input of spectrogram with 256 frequency bins and 500 timesteps,

The figure on the left is the accuracy of baseline model after 4 epochs. We can see that although the training accuracy is on a upward trends, the validation accuracy starts to decrease after 2 epochs, with a peak at 26%.

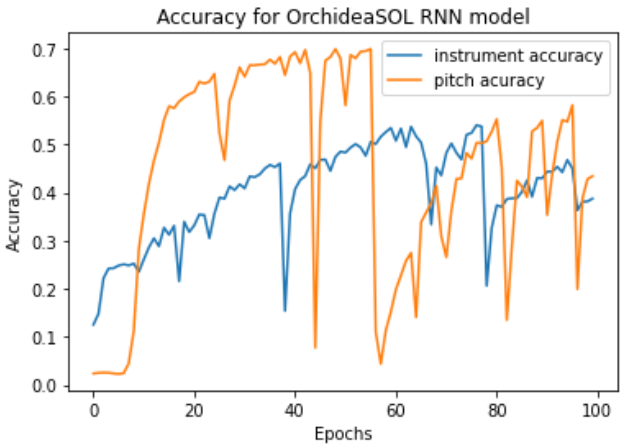
CNN with 2 convolutional layers

This model is similar to baseline model, with 2 convolutional layers instead of 1. The training time for this model has a dramatic increases to 30 minutes per epoch. Hence only a short evaluation was performed.



The accuracy for both the training and validation starts decreasing after 3 epochs, and the accuracy only peaks at 15%. After consideration, we can make the decision to move on to the next RNN/LSTM model, since the short evaluation indicated that CNN might need more hyperparameter tuning and training time to be able to perform well.

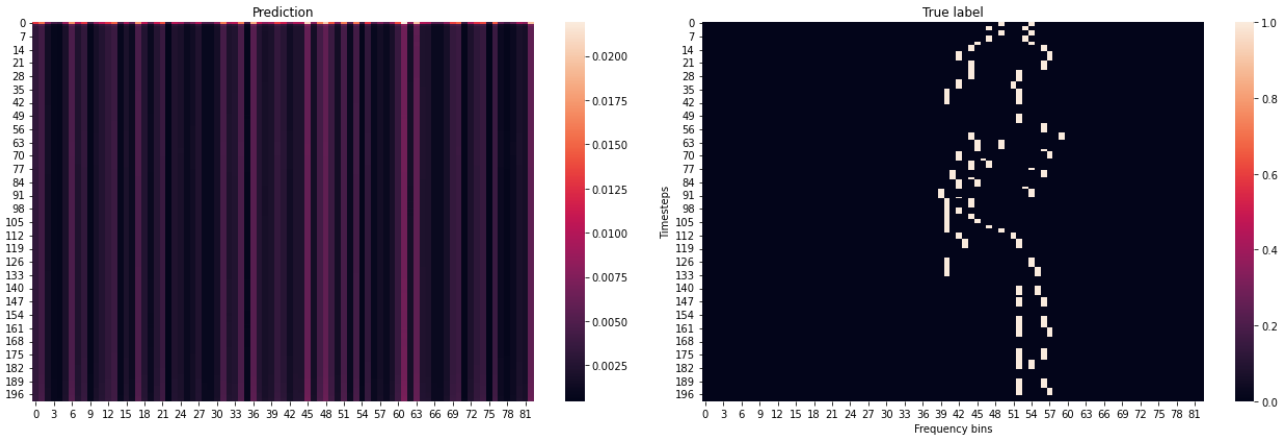
RNN/LSTM



Since our spectrograms are a sequential data, it is only logical to consider RNN (recurrence neural network)/LSTM (Long short term memory) for our models consideration. RNN is able to capture the relation between sequence, as the model takes the output of previous sequence to combined with the next input, in order to make a prediction. We have also defined the architecture to classify both the instruments and notes.

Although we have had good results for accuracy, weird behavior starts to happen around 55 epochs, we have a massive drops in notes accuracy, with periodic drops for both accuracies arond 10 epochs. The best results is located in around 70 epochs, with both accuracy peaked around 50%. A more complex model with convolutional layers feeding to LSTM is proposed.

Unrolled LSTM



Left: Prediction of unrolled LSTM model on notes as x-axis and timesteps as y-axis. Right: True label.

Current progress on sequential unrolled LSTM is horrible to say the list, more than 10 variations of architecture had been tested, including the pipeline of convolutional layers, embedding layers to LSTM layers had been tested. Simpler architect of 2 LSTM layer to classification had also been tested, however, the models always converged to predicting constant value across timesteps, despite the custom loss function which emphasis on reducing false negatives. Further works needed to be done with checking the preprocessing and data generating pipeline.

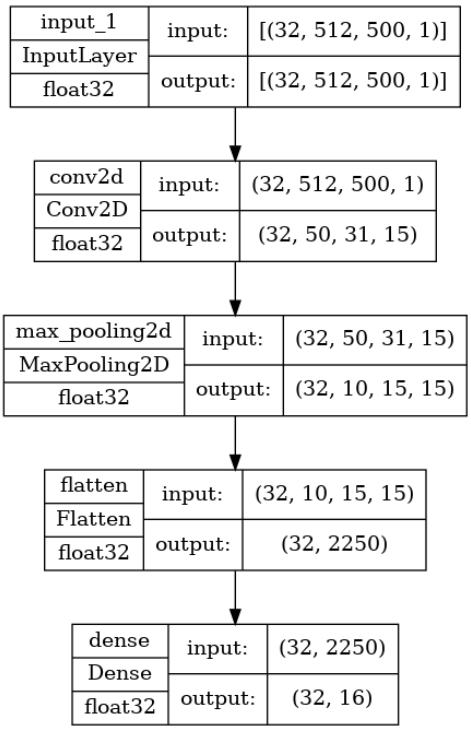
Conclusion

Despite the potential of LSTM model on music transcription, the project is stranded at a point where significant time are needed to tune the architecture to be functioning. However, we have demonstrated that on single instrument and single note classification, the LSTM model functions reasonably well. Future works include examining the cause of the behavior of LSTM model, and optimize hyperparameters. Also, weighted cross entropy did'n't work as desired, need to fix bias in dataset.

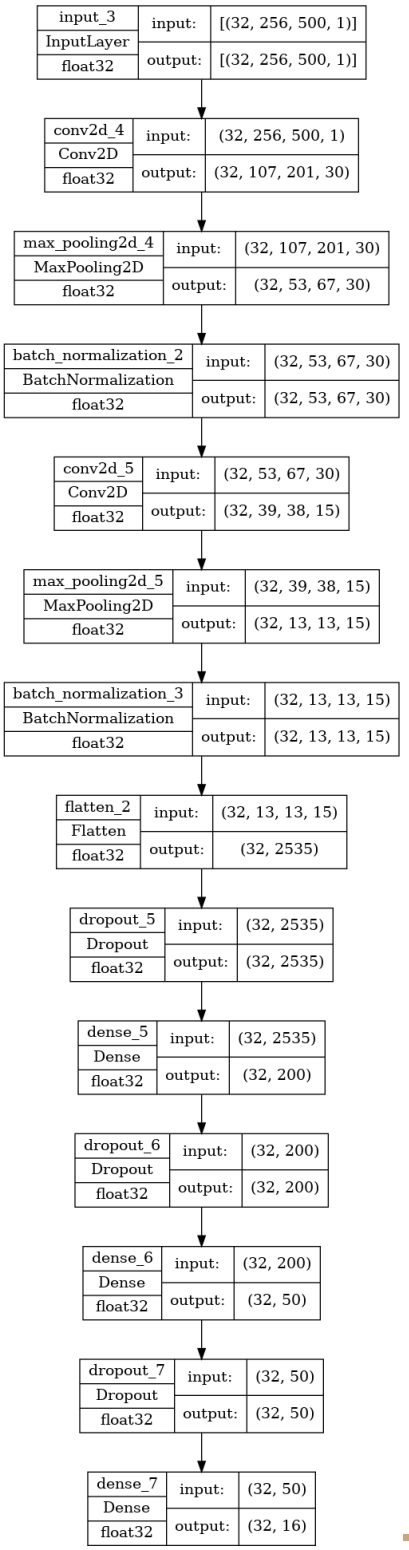
Appendix

Neural network Architecture

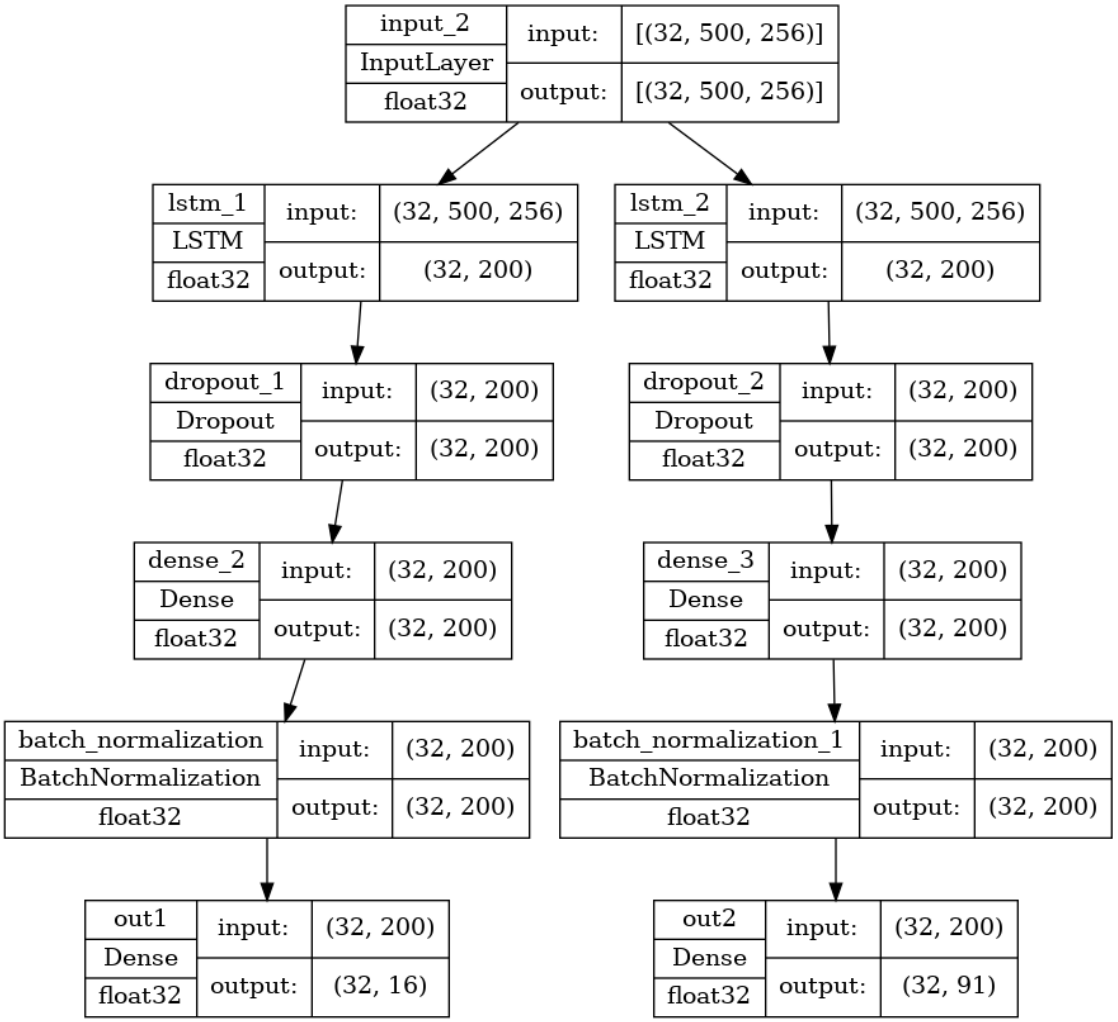
Baseline



2 Conv CNN



RNN/LSTM


input_2
InputLayer
float32
input:
output:
[(32, 500, 256)]
[(32, 500, 256)]
lstm_1
LSTM
float32
input:
output:
(32, 500, 256)
(32, 200)
lstm_2
LSTM
float32
input:
output:
(32, 500, 256)
(32, 200)
dropout_1
Dropout
float32
input:
output:
(32, 200)
(32, 200)
dropout_2
Dropout
float32
input:
output:
(32, 200)
(32, 200)
dense_2
Dense
float32
input:
output:
(32, 200)
(32, 200)
dense_3
Dense
float32
input:
output:
(32, 200)
(32, 200)
batch_normalization
BatchNormalization
float32
input:
output:
(32, 200)
(32, 200)
batch_normalization_1
BatchNormalization
float32
input:
output:
(32, 200)
(32, 200)
out1
Dense
float32
input:
output:
(32, 200)
(32, 16)
out2
Dense
float32
input:
output:
(32, 200)
(32, 91)

Convolution to Unrolled LSTM