CSCI4999 Final Year Project

2018 - 2019 Second Term Report

Automatic Piano Reduction (Backend II): Algorithms

KY1802

Szeto Ethan (1155079061)

Yik Wai Pan (1155079854)

Department of Computer Science and Engineering

Faculty of Engineering

The Chinese University of Hong Kong

Supervisor: Prof. Kevin Yip Yuk Lap

Co-supervisor: Dr. Lucas Wong

April 2019

Contents

1. Abstract

The piano reduction project is the continuation of a multi-year project. Using machine learning, musical pieces are reduced to a two-hand staff score that resembles the original score and is playable by a single pianist.

Continuing the progress from the last semester, recurrent neural network models are studied and used to apply the reduction on the musical pieces. Contrary to the work done by previous years' groups, a binary representation is used to describe the played notes and allows more complicating transformation of the original music score. As a result, apart from simply keeping or discarding notes that are in the original score, the machine learning component is also capable of adding extra notes that match the reduction style.

On top of that, models that consider a wider range of adjacent notes are built in this semester to further improve the accuracy. Extending from the model proposed in the last semester, the notes that are played in both previous and subsequent moments can now be utilized by a Gated Recurrent Unit model.

The methods and evaluations of learning a specific reduction style are also explored in this project. By using the transpositions of a given reduction score, the aforementioned models are capable of imitating the style and achieve better performance than the baseline models. Different factors, such as the chord feature and the input length of the models, are also examined in the project and show little influence on the performance of the machine learning components.

Furthermore, a post-processor is also implemented in this project. Considering different factors like the hand movements and busyness, penalties are defined to express the playability of a musical score. Hill climbing, a heuristic approach, is applied in the post-processor to transform the reduced musical score into a more playable score.

2. Introduction

2.1. Project Background

This piano reduction project has started since 2013. It aims to simplify a multi-staff score of a symphony or an orchestra performance into a two-staff score. After the reduction, the resulting piece should resemble the original score and be playable by a single pianist.

The project involves learning the reduction style and transform the given musical piece by deciding which notes should be kept or not, and at the same time adding or adjusting notes based on the style. Aside from performing reductions using complex musical knowledge, this project takes a computer science approach to solve the problem.

2.2. Terminologies

For the ease of communication, the following are the terminologies that are used in the report.

Terminology	Definition
Original Score	The score before performing piano reduction. It consists of all notes played by all instruments.
Answer Score	The score after piano reduction is performed by human. Most of them are obtained from the Internet.
Reduced Score	The score generated by the machine learning component.
Processed Score	The score generated by the post-processor after the score is reduced by the machine learning component.
Moment	A moment of the score which one or more note(s) are to be played.

2.3. Motivation

The piano reduction model developed by Yiu and Choi (2017) has its limitations. Determining whether the notes in the original score are kept or not only, the model was not capable of considering new notes in the answer score and these new notes could not be generated to the reduced score. Compared with that of a dummy model that keeps all notes in the input score, the numerical performance of their model could not surpass the dummy model and the model cannot achieve satisfactory result.

Pairwise CRF	Accuracy	Δ accuracy				
Trained on	Dummy	Test #1	Test #2	Test #3	Test #4	Training
#1 - Beethoven Symphony No. 5, Mov. 1	0.610821		0.040418	-0.00717	-0.063885	0.075923
#2 - Beethoven Symphony No. 7, Mov. 2	0.648644	-0.078475		-0.032063	-0.057675	-0.004933
#3 - Dvorak New World Symphony No. 9 Mov. 2.xml	0.675892	-0.086026	-0.028405		-0.04807	0.046978
#4 - Cosi fan tutte Act 1 Scene XI	0.650458	-0.098729	-0.007059	-0.023268		0.017516
				Mean Δ	-0.0408673	

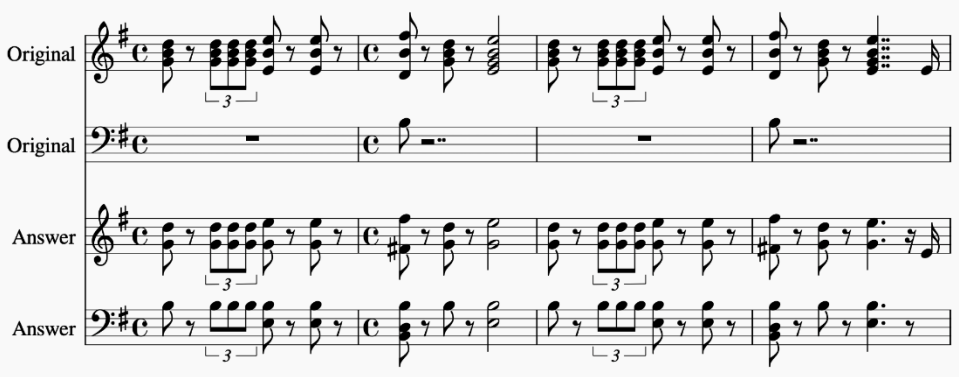
Independent Onsets	Accuracy	Δ accuracy				
Trained on	Dummy	Test #1	Test #2	Test #3	Test #4	Training
#1 - Beethoven Symphony No. 5, Mov. 1	0.610821		0.04824	0.007823	0.03781	0.079531
#2 - Beethoven Symphony No. 7, Mov. 2	0.648644	-0.022197		-0.068109	-0.041169	0.002845
#3 - Dvorak New World Symphony No. 9 Mov. 2.xml	0.675892	-0.038237	-0.034596		-0.028769	-0.010561
#4 - Cosi fan tutte Act 1 Scene XI	0.650458	-0.008105	-0.016471	-0.020654		0.029542
				Mean Δ	-0.0153695	

Onset Chain	Accuracy	Δ accuracy				
Trained on	Dummy	Test #1	Test #2	Test #3	Test #4	Training
#1 - Beethoven Symphony No. 5, Mov. 1	0.610821		0.079531	-0.076271	-0.086049	0.092569
#2 - Beethoven Symphony No. 7, Mov. 2	0.648644	-0.009486		-0.141909	-0.166572	-0.037944
#3 - Dvorak New World Symphony No. 9 Mov. 2.xml	0.675892	-0.019301	-0.026584		-0.142025	-0.109614
#4 - Cosi fan tutte Act 1 Scene XI	0.650458	0.009411	0.022745	-0.055164		-0.072157
				Mean Δ	-0.0509728	

Figure 11.1: Evaluation results for our three models. Red indicates worse performance compared to the dummy model, and green indicates better performance compared to the dummy model.

[7] Results of last year's model

It can be seen from the reductions collected in the past that there are a number of newly added notes that are not in the original score. For example, the following reduction of Nutcracker March has extra notes in the bass clef. If the previous method is kept, a high accuracy is unlikely to be achieved. By learning the invented notes in the answer score, it can be expected that the performance of the model can be greatly improved.



A segment of reduction of Nutcracker March

More importantly, the reduction style could not be evaluated in the past. A score can be reduced in different ways based on different styles, and a good learning method should be able to train different models that fit different styles. As the aforementioned evaluation done by Yiu and Choi (2017) used different scores with different reduction styles as training and testing data, the results could not tell whether the models effectively learn the reduction style.

On top of that, it is difficult to find training samples with both original score and answer score, the number of usable musical piece is insufficient. Moreover, since those training examples are downloaded online and they were done by separate contributors, their reduction styles are different for most of them, making the amount of training examples for each reduction style too small. Therefore, it is challenging to understand the rules which the authors adopted to perform the reduction and investigate the learning of the reduction style.

If a musical piece that exists multiple versions of reduction is found, the effectiveness of learning the reduction style can be better evaluated. Having a number of reductions of Mozart's Cosi fan tutte, we are able to further investigate the problem in this semester.

2.4. Objective

In this semester, our piano reduction project aims to work on the followings:

- To further improve the machine learning component implemented in the first semester

- To facilitate the machine learning of reduction styles from the training samples
- To investigate the usefulness of the chord feature
- To implement a post-processor to increase the playability of the score reduced by the machine learning component
- To refine all the components into a reusable package, users can call our components by functions programmatically

2.5. Work Distribution

2.5.1. Szeto Ethan (1155079061)

- Produce examples for investigation of chord feature
- Design and implement the piano playability calculation model
- User-configurable playability optimization algorithm (Hill Climbing)
- Analysis on the reduction results

2.5.2. Yik Wai Pan (1155079874)

- Design and implement the pre-processor to facilitate the reduction
- Design and implement the machine learning component using the Keras library
- Perform experiment using different machine learning models
- Evaluate the effectiveness of learning reduction styles and chord features

3. Overview of Last Semester

3.1. Architecture Overview

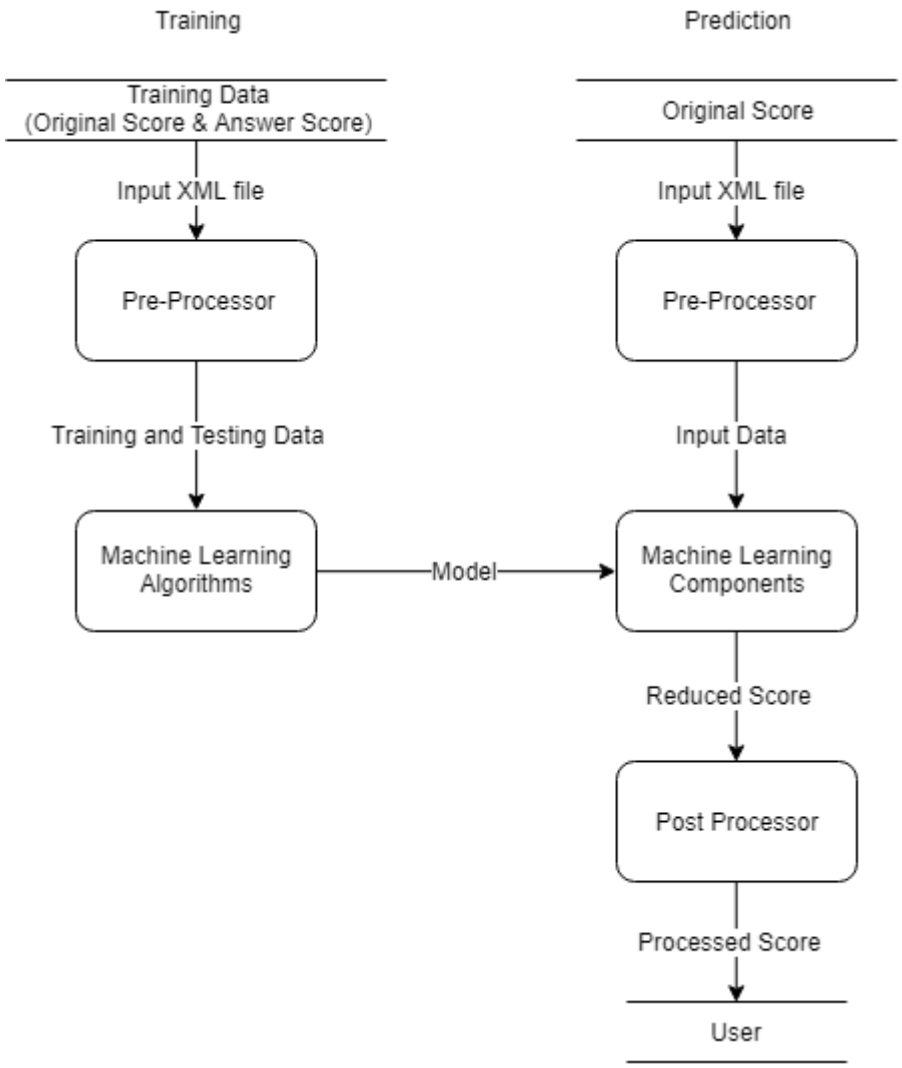


Diagram of the System Architecture

The architecture follows the approach from the last year's group. The system is divided into three parts, namely the pre-processor, machine learning component and post-processor. It takes the original score in MusicXML format as input, and then output a playable reduction that matches the required reduction style. However, the machine learning models will be too complex if it considers both the reduction style and the playability. Therefore, the system is divided into parts to simplify the machine learning component.

The piano reduction system uses MusicXML file as input. To predict the piano reduction of the input piece, the file will be processed by the three components through the following procedures.

The pre-processor converts the MusicXML file into data usable by the machine learning component. The file in MusicXML format stores all elements of the score, such as notes, key signatures, tempo etc. Although it completely describes the musical piece, the notes are not well organised and cannot be manipulated easily. Therefore, the pre-processor first performs instrument transposition to shift the notes from their written pitch to their concert pitch. Then all notes are extracted from the score and have their musical features computed. They are all stored as a Dataframe so that the notes can be accessed by the following components conveniently. In addition, labels are added for each note to facilitate the machine learning, piano reduction and post-processing.

The machine learning component reduces the input score using the given pre-trained model. Having learnt the reduction style from the training data, the component determines which notes should be kept, discarded or added based on the learnt style. It also allows users to generate machine learning models of a specific reduction style using the original and answer score of a music piece.

Lastly, the post-processor further adjusts the predicted reduction score and generates a more playable score. By considering several musical factors, such as hand movements, finger positions etc., the score would have some notes changed to ensure that it can be easily played by a single pianist. The score would then be converted back to a MusicXML file for users.

3.2. Machine Learning Component

In the past few years, different models are proposed for the machine learning component to perform the piano reduction task. Ng (2014) used a dense neural network to determine whether the notes should be kept, and Choi and Yiu (2017) introduced Conditional Random Field last year to allow considerations of adjacent notes. However, these models only decide which notes in the original score should be remained in the reduction. When the reduction has additional notes or shifts the

notes by octaves, the models proposed in the past fails to generate any of them, causing a low accuracy as the result.

3.2.1.Binary Representation of Notes

Attempting to improve the accuracy, we have used an alternative way to represent the notes. For each moment that has notes in the input, the notes can be described by a binary vector of length 128, as all pitches are denoted by an integer from 1 to 128. The i^{th} value of the vector denotes whether there is a note with pitch of i started at that time (1 represents yes, and 0 otherwise). For example, if there is only a chord C4-E4-G4 (pitches are 60, 64, 67) at one time, the binary vector will be:

$$x_i = \begin{bmatrix} (59\ 0's) \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ (61\ 0's) \end{bmatrix}$$

As mentioned before, if we only decide whether a note in the original score should be kept or not, the notes in the answer score that are not contained in the original score can never be captured in the reduced score. By using this representation, the model can even add new notes that did not appear in the input score. Whenever a value in the vector changes from 0 to 1 after the prediction, a corresponding note can be added into the score starting at that time.

The musical features of each note introduced by the previous groups are used. For example, whether the note belongs to the active rhythm parts and whether the note belongs to the pitch class that appears the most are marked and used as binary features for each note. The features are then integrated into the binary representation of the notes in order to provide the musical properties of the occurred notes to the machine learning model. The value of features of all existing notes are appended to the binary vector. The new input vector has the length of $128 * (m + 1)$. For each note with pitch i, the m features computed are positioned at the $(129 +$

$(i-1)*m)$ to the $(128+i*m)$-th value of the vector. If the note with pitch i absents at that time, the values will be replaced by 0 instead.

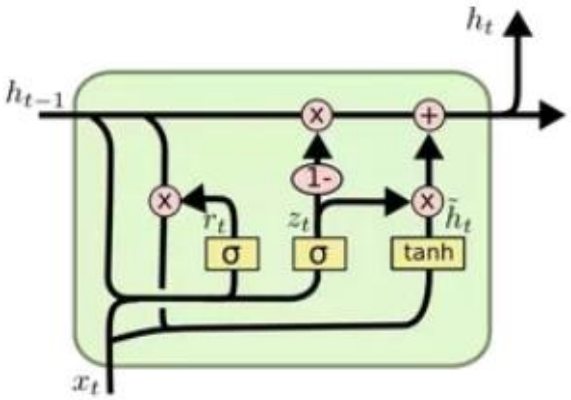
$$x_{i_j} = \begin{cases} 1, & Note\ with\ pitch\ j\ presents\ at\ that\ offset\ time \\ 0, & Otherwise \end{cases}, 1 \le j \le 128$$

$$x_{i_{128+(m-1)*j+k}} = \begin{cases} the\ k^{th}\ feature\ of\ note\ with\ pitch\ j, & Note\ with\ pitch\ j\ presents\ at\ that\ offset\ time \\ 0, & Otherwise \end{cases},$$
$$1 \le j \le 128, 1 \le k \le m$$

With the help of the features, the models will be capable of predicting the labels using the musical properties.

3.2.2.Use of Recurrent Neural Network

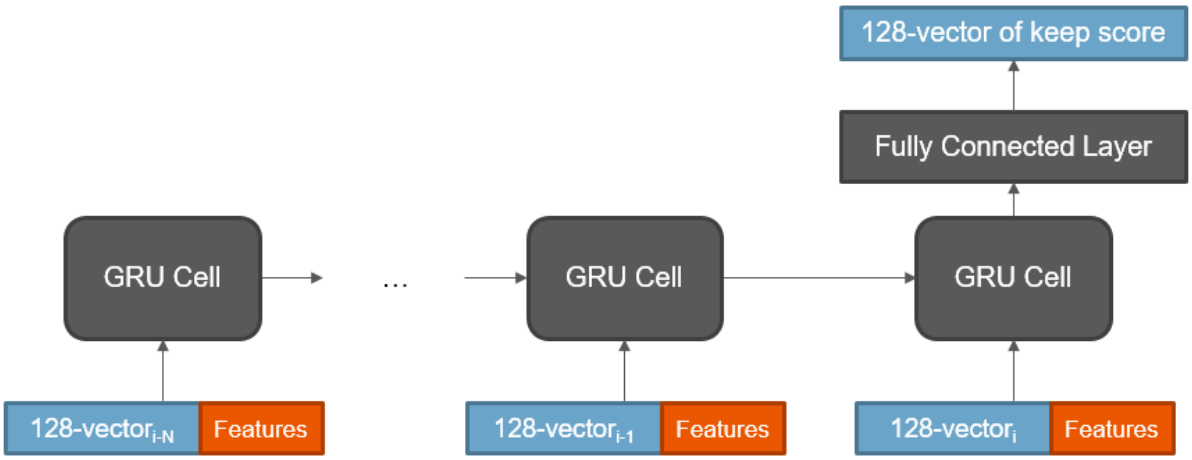
In addition, we have introduced the Long-Short Term Memory model (LSTM) and the Gated Recurrent Unit model (GRU). A special type of Recurrent Neural Network (RNN), LSTM overcomes the long-term dependencies problem, which the gradients of the distant information vanish during back-propagation process, by introducing the concept of gates. Consisting of an input gate, an output gate and a forget gate, the LSTM model has the ability to make predictions based on a longer series of information. It is because the information stored in the cell state can be transferred from the previous cells to the current cell. Similar to the LSTM unit, a GRU unit is a simplified version that consists of two gates only. Due to the fewer number of parameters, the GRU model is more computationally efficient and the overfitting problem could be alleviated.



[3] A Gated Recurrent Unit Cell

After observing the training samples, we believe that the information of the previous notes may also be useful to determine the label of one note. The label of previous notes may affect the prediction of the current note. Also, as mentioned above, LSTM and GRU can retain some long-term information. Therefore, we have the following approach.

To determine the output notes of a specific moment, the previous N moments are also taken into consideration. The N moments are represented by N binary vectors described in the previous part, and the sequence of binary vectors compose a matrix that is used as the input of the Recurrent Neural Network model. The model then outputs a 128-vector that shows the likelihood of having each note in the reduction.



The machine learning model proposed in the last semester

3.3. Comparison of Models

Multiple metrics were used in the evaluation stage, namely Accuracy, F1 Score, Jaccard Similarity and ROC AUC Score. The first three metrics evaluate the performance of the selection of notes based on a given threshold, and the last one measured the performance of the model on distinguishing which notes should be played in the reduced score regardless of the threshold. Foreseeing that the post-processor might change some notes by octaves to maintain playability, Pitch Class

Jaccard Similarity was also introduced to calculate the similarity considering only the pitch class of the notes at the same offset.

Two experiments were performed to assess the performance of the proposed machine learning models. The first one consisted of a cross validation using the Symphony No. 5 In C Minor of Ludwig van Beethoven, and the other one trained the model using a number of musical pieces in order to obtain a general reduction model.

The Symphony No.5 is split into 5 parts evenly. In the i^{th}-fold run, the i^{th} part of the score is selected as the validation data and the models use the rest of the parts as the training data. The testing metrics are then computed using a threshold of 0.5, and the following table shows the averaged results over all 5 runs.

Model	AUC ROC Score
Neural Network	0.94468249
Neural Network with features	0.94109897
LSTM	0.95534761
LSTM with features	0.95058434
GRU	0.95843161
GRU with features	0.95130540

To train a more general piano reduction model, several scores are used as the training data. The aforementioned models are evaluated by using the following scores to train:

- Beethoven's Symphony No. 7 Movement 2
- Mozart's Symphony No. 25
- Mozart's Symphony No. 40
- Dvorak New World Symphony No. 9 Movement 2
- Beethoven's Symphony No. 7 Movement 1

And then they are validated using Beethoven's Symphony No. 5 Movement 1. The AUC ROC Score of the models are as follows:

Model	AUC ROC Score
Neural Network	0.9511436782152689
Neural Network with features	0.9650616296621177

LSTM	0.972017474424736
LSTM with features	0.9851146881592856
GRU	0.973463969797602
GRU with features	0.9863432431011727

The result suggested that the Gated Recurrent Unit model has better performance than the Long Short-term Memory model and the dense neural network model. All metrics indicated that the GRU model could generate a more reasonable reduction than the other models. The usefulness of the musical features remained unknown, as the two evaluations gave opposite results when musical features were added. A possible cause of this might be the insufficient amount of training data. Therefore, the musical features will continue to be used by the machine learning models in this semester.

4. Machine Learning Component

4.1. Introduction

In the last semester, the objective of constructing the machine learning component is mainly to generate a reasonable reduced score that is playable by a single pianist and resemble the original score at the same time. The original and answer scores of multiple pieces were then all used as training data to train the Recurrent Neural Network models. While the result showed that the models were able to generate such reduced scores, a particular reduction style could not be learnt by the model, which was trained by scores with different reduction style.

In this semester, we investigated the methods of generating models for a particular style and evaluated the effectiveness of different models. To generate such models, only one original score and its corresponding answer score are used to train the models. Three answer scores of Mozart's Cosi fan tutte, each with its own reduction style, were used to train and study how well the models can capture the styles.

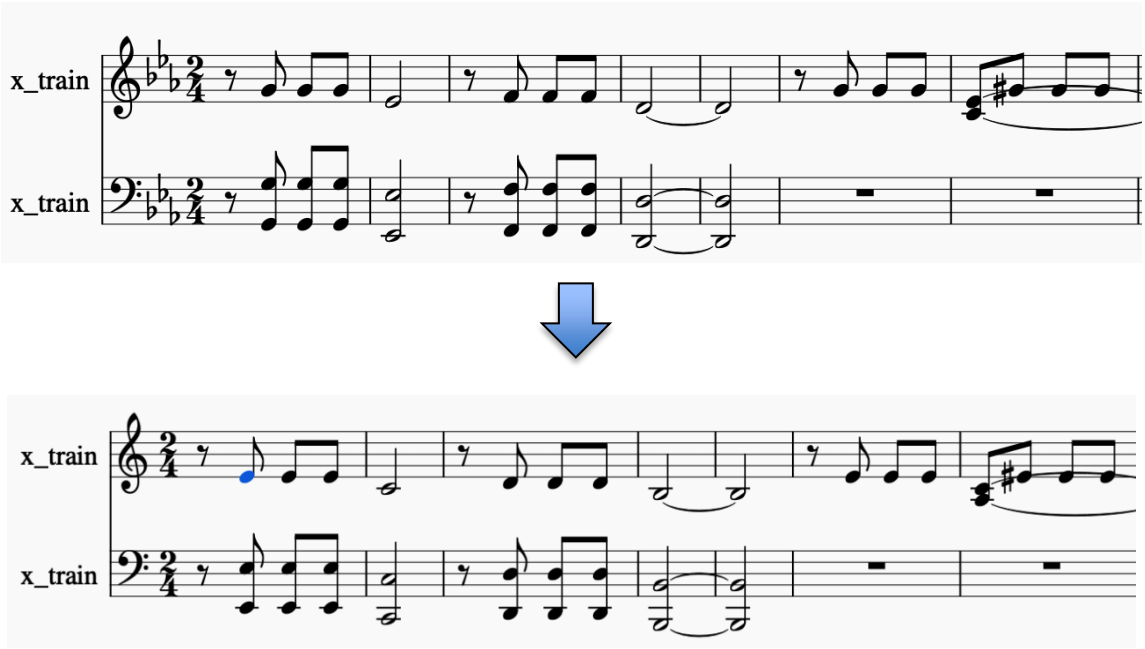
4.2. Generating Models with Particular Reduction Style

Currently the models simultaneously consider the notes with same offset and return the reduced notes combination for that offset. However, simply using the original data may not be sufficient. For instance, some notes may not appear in the original score. The chords that are in different keys with the original score could also never appear in the training data, making the model not be able to output the transposed chords accordingly. Therefore, the following two approaches were proposed and experimented.

4.2.1. Key Standardization

Before the score is input into the model, key transposition is first performed on the score. The whole musical piece is transposed to C Major or A Minor, which has no flats and no sharps. For example, if the original piece is in E Major, which is 4 pitch class higher than C Major, all notes will be shifted down by 4 pitch classes. Likewise, a B Minor score will have all notes shifted down by 2 pitch classes. After the model

generates the reduction, the reduction is transposed back to the original key reversely to retrieve a score that matches with the original input.



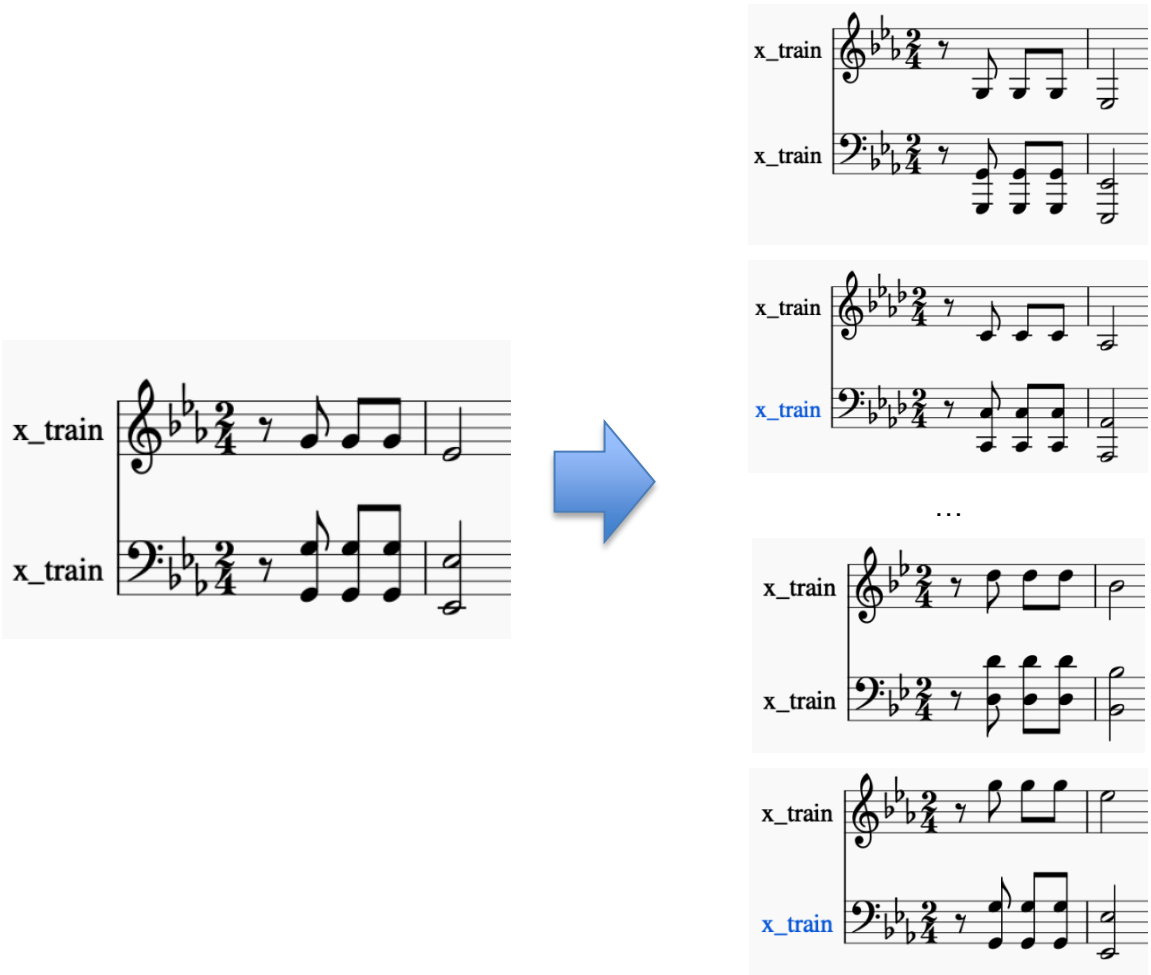
Example of standardizing the score by shifting all notes down by 3 pitch classes

This approach ensures that all scores input into the models will have no sharps and no flats in their key. The chance of having unseen notes and chords would be lowered, making the model easier to generate a reasonable reduction.

In practice, the approach faced several difficulties and was not capable of generating reasonable reductions when different musical pieces were input into the model. One of the problems is that the insufficient size of training data causes the models output notes that are highly similar to the training data. Another problem is shown when modulation occurs in the musical pieces. The musical pieces often modulate to a closely related or parallel key that may or may not be accompanied by a change in their key signature. These modulated parts would not be able to be standardized to C Major or A Minor if the score is transposed based on its initial key signature. Shifted to another unknown key, the notes in the modulated parts would not be reduced by the models correctly due to the lack of samples.

4.2.2.Key Transpositions of Training Data

Another approach is to increase the variety of training data by generating multiple versions of it. Before the original and answer score are used to train the model, 25 copies of the scores are generated, with notes in each score shifted by -12 to +12 pitch classes respectively. The copies are then all used to train the models.

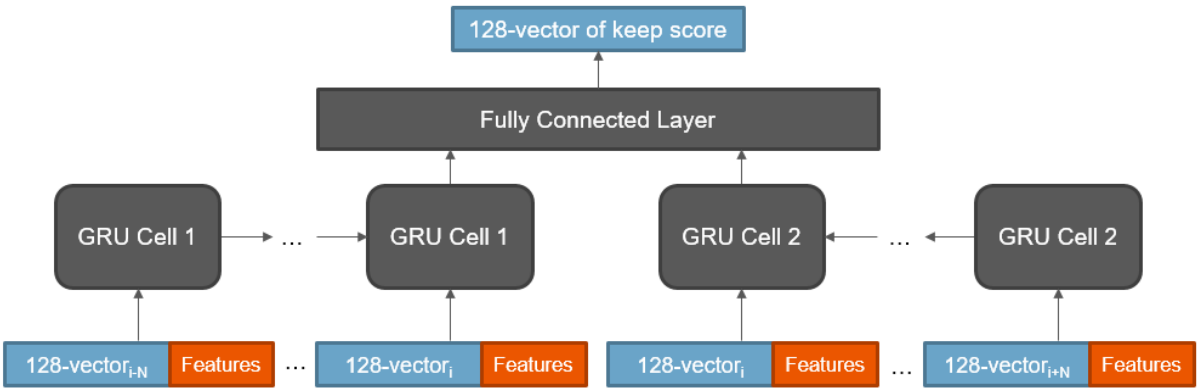


Transposing the training data into different keys

Contrary to standardizing the input using the initial key signature, this approach can handle musical pieces with modulation properly. The increased number of training data allows the models to reduce different kinds of notes and chords regardless of their keys and tones. As a result, the models generate more reasonable reductions using this approach.

4.3. Considering Both Sides of Adjacent Notes

The model proposed in the last semester predicts the notes at a specific moment by considering all the notes occurred in previous moments. Using a Gated Recurrent Unit network, the model used the binary representations of previous moments as input to generate reductions. In addition to this, it is found that the notes in subsequent moments could also be useful to predict the reductions. Similar to the previous moments, the subsequent moments might provide helpful information, such as consecutive patterns and repeating notes, to improve the accuracy of the machine learning components. Therefore, a new model is proposed in this semester that takes both previous and subsequent moments into consideration.



The machine learning model using two Gated Recurrent Units

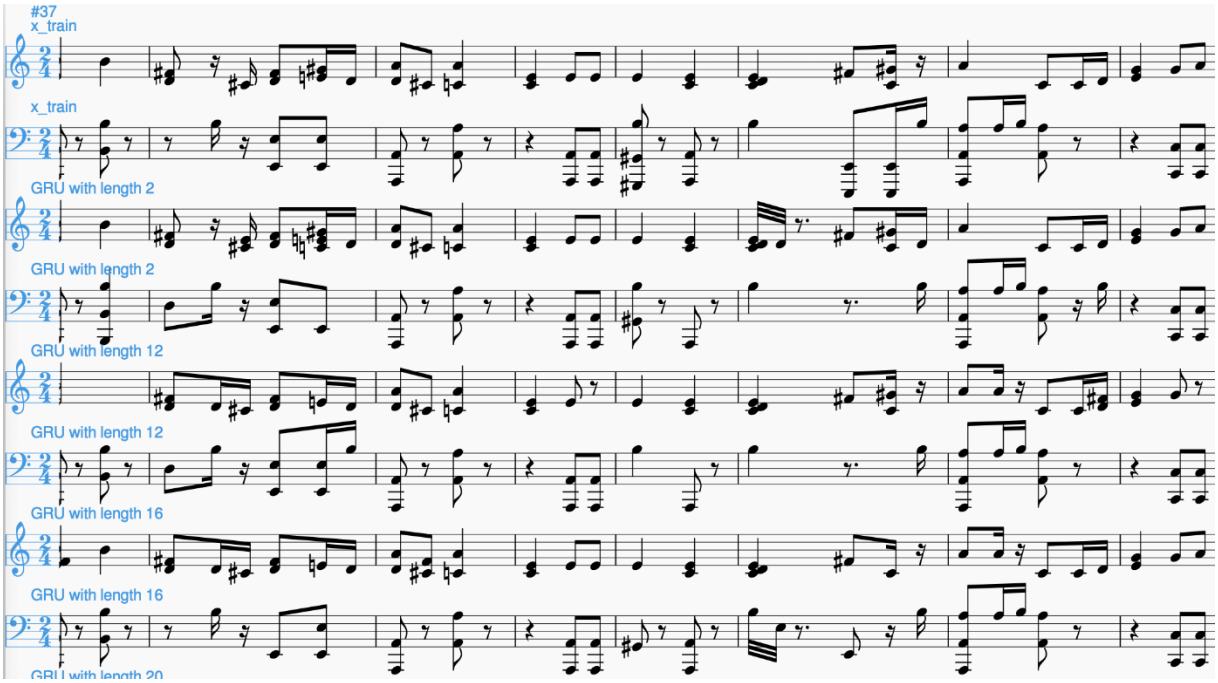
The new model consists of two Gated Recurrent Unit cells. Same as the one proposed in the last semester, the first cell takes the binary vectors of the current and previous moments as input. The second cell takes the current and subsequent moments as input reversely. That is, the binary vector of the i+N-th moment is input into the cell first, then that of the i+N-1-th moment is taken, and so on, until the i-th moment is read into the GRU cell. The output of the two cells is then concatenated and input into a fully connected layer. At last the layer generates a vector of 128 numbers that indicates the likelihood of having each note in the reduction. Same as the previous models, a note is decided to be shown in the reduction if the likelihood reaches a certain threshold.

4.4. Note Duration as Feature

Under the binary representation of the notes, the machine learning models treat each moment as the same. A moment of 4 beats and an moment of 1 beat would be considered and reduced in a similar way by the models. To overcome the problem, the note duration is added as a musical feature of the notes in order to help the model distinguish the differences and improve the performance.

4.5. Model Stacking

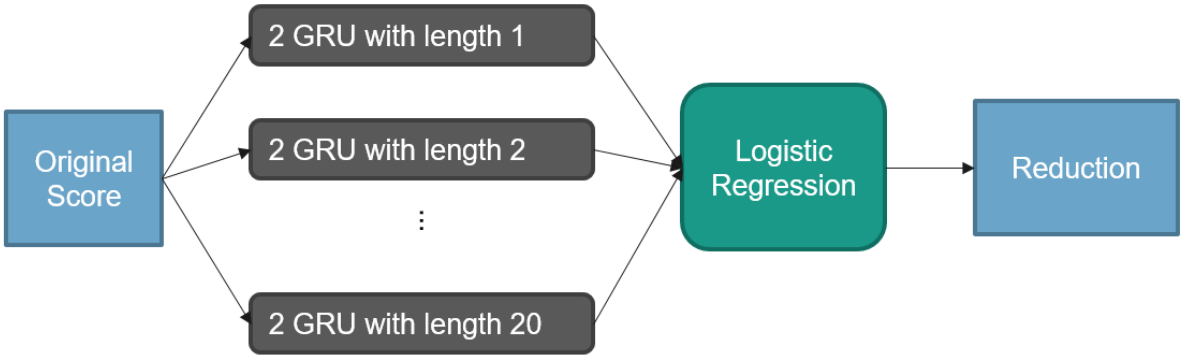
The aforementioned model takes a specific length of inputs to perform the reduction. In other words, a number of models can be generated with their own input length. Based on the tempo and the combination of notes, different models may behave differently and generate diverse reductions.



Different Reductions Generated by Models with Different Input Lengths

These Gated Recurrent Unit models can be combined as one single model to generate the reduction. For each note, all the likelihoods computed by models with different input lengths can be used as features to determine whether it should be included in the reduction. Together with other general features like duration of that

moment and offset, these likelihood features can be used to train another classification model to generate a final reduction.



The Stacked Model using outputs of GRU models as features

In practice, the stacked model performed poorer than simply using a Gated Recurrent Unit model. Extra data is needed to train the final stacking classifier, which causes a drop of data size for training the Gated Recurrent Unit models. The drop in accuracy of the GRU models cannot be mitigated by model stacking, making the outcomes of the stacked model unsatisfactory.

4.6. Evaluation

4.6.1. Dataset

Mozart's Cosi fan tutte and Tchaikovsky's Nutcracker March are used in the evaluation section.

Scores of the Mozart's Cosi fan tutte are provided by Dr. Lucas Wong. A total of 9 versions of reductions are obtained. In the following section, three versions of the reductions are used, namely Cosi 2, Cosi 7 and Cosi 9. Cosi 2 is published by Robert Birchall in 1809, and Cosi 7 is the one reprinted by Edwin F. Kalmus in 2000. Cosi 9 is the one reduced by Dr. Lucas Wong.

Scores of the Tchaikovsky's Nutcracker March are obtained from the previous group. There is only one version of reduction.

4.6.2. Evaluation Metrics

The machine learning component generates likelihoods of 128 notes for each moment, indicating how likely the notes should be in the reductions. However, most notes will not be played in most cases. For instance, the notes that have low or high pitch would seldom be played. The notes that cannot form chords with input notes would be unlikely to be added as well. Therefore, to effectively indicate the performance of the reduction models, only the following will be counted during the evaluation:

- Notes shown in the Original Score
- Notes shown in the Answer Score
- Notes shown in the Reduced Score

Three metrics are used to evaluate the performance of the models, namely F1 Score, Accuracy and Jaccard Similarity. Defined as $\frac{number\ of\ correct\ labels}{total\ number\ of\ labels}$, the Accuracy can serve as a simple indicator of how the reduced score matches with the answer score. F1 Score is defined as the harmonic mean of precision and recall, which is

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

, where Precision is the percentage of correctly kept notes among all kept notes in the prediction, and Recall is the percentage of correctly kept notes among all notes that should be kept in the answer score. Since the answer score is a biased dataset that has fewer positive labels than the negative ones, the F1 Score is used to show the correctness of the chosen notes in the score reduced by the machine learning component. Lastly, Jaccard Similarity is defined as $A_j = \frac{|X \cap Y|}{|X|+|Y|-|X \cap Y|}$, where X is the answer score and Y is the reduced score. The machine learning models may generate extra notes that are not in either the correct score or the original score. As every difference between the answer score and the reduced score lowers the Jaccard Similarity, it becomes a useful metric that indicate whether the newly generated notes match with the answer score.

Contrary to last semester's evaluation, changes of octaves are valued as it plays an important role in the reduction style. The performance of predicting pitch classes only is not evaluated in this semester.

4.6.3. Effectiveness of Learning the Reduction Style

To study how well the models can capture the styles, three answer scores of Mozart's Cosi fan tutte, each with its own reduction style, were used to evaluate the models. Two of reductions (Cosi 2 and Cosi 7) have a large number of notes, and the other one (Cosi 9), reduced by Dr. Lucas Wong, has less notes and is more playable.

The Gated Recurrent Unit models proposed in both semesters are evaluated. Three models are also used as baseline models, including logistic regression classifier, naïve bayes classifier and keeping all notes from the Original Score.

To ensure that the testing data have enough previous and subsequent moments, 10% of the data is extracted from the middle part of the Answer Scores to be used as testing data, and the remaining 90% of the three Answer Scores is used to train their own models respectively.

Then the 10% testing data of the three reductions is input into different types of machine learning models that are trained by different reductions. For each type of models, it is expected that if the training data of the model is the same reduction with the testing data, the model will achieve higher scores than the others.

The following tables show the performance of each model when different reductions of Mozart's Cosi fan tutte are input as testing data.

Testing Data: Cosi 2				
Model	Training Data	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Cosi 2	0.80076628	0.75644028	0.66773163
	Cosi 7	0.7609075	0.67915691	0.61408451
	Cosi 9	0.69201521	0.6206089	0.52906977
GRU considering both previous and subsequent moments	Cosi 2	0.85992218	0.83138173	0.75426621
	Cosi 7	0.77205882	0.70960187	0.62874252
	Cosi 9	0.73387097	0.69086651	0.57961783
Logistic Regression	Cosi 2	0.71428571	0.65339579	0.55555556
	Cosi 7	0.74545455	0.67213115	0.5942029
	Cosi 9	0.54590571	0.57142857	0.37542662
Naive Bayes	Cosi 2	0.56763926	0.61826698	0.3962963
	Cosi 7	0.5655527	0.60421546	0.39426523
	Cosi 9	0.54068242	0.59016393	0.3705036
Keep All	/	0.76071429	0.68618267	0.61383285

Testing Data: Cosi 7				
Model	Training Data	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Cosi 2	0.73743017	0.66978923	0.5840708
	Cosi 7	0.82312925	0.75644028	0.69942197
	Cosi 9	0.74676525	0.67915691	0.59587021
GRU considering both previous and subsequent moments	Cosi 2	0.78638941	0.735363	0.64797508
	Cosi 7	0.86940966	0.82903981	0.76898734
	Cosi 9	0.75146771	0.70257611	0.60188088
Logistic Regression	Cosi 2	0.72795497	0.66042155	0.57227139
	Cosi 7	0.75752212	0.67915691	0.60968661
	Cosi 9	0.57416268	0.58313817	0.40268456
Naive Bayes	Cosi 2	0.52040816	0.55971897	0.35172414
	Cosi 7	0.51980198	0.54566745	0.35117057
	Cosi 9	0.47474748	0.51288056	0.31125828
Keep All	/	0.77217391	0.69320843	0.62889518

Testing Data: Cosi 9				
Model	Training Data	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Cosi 2	0.68852459	0.64402810	0.52500000
	Cosi 7	0.72727273	0.65573771	0.57142857
	Cosi 9	0.75609756	0.71896956	0.60784314
GRU considering both previous and subsequent moments	Cosi 2	0.75416667	0.72365340	0.60535117
	Cosi 7	0.74901961	0.70023419	0.59874608
	Cosi 9	0.80952381	0.79391101	0.68000000
Logistic Regression	Cosi 2	0.66942149	0.62529274	0.50310559
	Cosi 7	0.67054264	0.60187354	0.50437318
	Cosi 9	0.56910569	0.62763466	0.39772727
Naive Bayes	Cosi 2	0.54810496	0.63700234	0.37751004
	Cosi 7	0.54084507	0.61826698	0.37065637
	Cosi 9	0.51296830	0.60421546	0.34496124
Keep All	/	0.68821293	0.61592506	0.52463768

The metrics indicate that the Gated Recurrent Unit models can learn the reduction style better. When the model with same reduction is used, significantly higher F1 Score, Accuracy and Jaccard Similarity can be achieved. Both GRU models outperform the baseline models, which give low scores no matter which training data is used.

Among the two Gated Recurrent Unit models proposed in these two semesters, the one that considers both previous and subsequent moments always attains higher

scores than the other one. This suggests that the subsequent moments are capable of providing useful information that helps generate the reductions.

4.6.4.Chord as Feature

The importance of chords for piano reduction could not be known in previous years, and in this semester, we added chord as musical features of the notes and evaluate their influence to the accuracy of the machine learning component. If the chord is capable of improving the performance of the machine learning component, chord identification can be done beforehand to enhance the models.

Cosi fan tutte and Tchaikovsky's Nutcracker March are used to perform the experiment. The chords given by Dr. Lucas Wong are manually input into the Original Score by marking which notes are in the chords. This becomes a binary feature for the notes that gives 1 if the note is in the chord, and 0 if not. With the addition of the new binary feature, new Gated Recurrent Unit models are trained to compare the results with the ones which do not use the chord as one of the features.

A 5-fold cross validation is done to evaluate the average scores of the models with and without the chord feature. The score is split into 5 parts and each time a part is used as the testing data, similar to the evaluation before.

The following are the results of the models:

Testing Data: Mozart's Cosi fan tutte				
Model	Using Chord as Feature?	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Yes	0.640748714	0.560737148	0.478001895
	No	0.641015116	0.557500050	0.478107646
GRU considering both previous and subsequent moments	Yes	0.648954164	0.576282305	0.487183272
	No	0.655681072	0.571184689	0.497043710

Testing Data: Tchaikovsky's Nutcracker March				
Model	Using Chord?	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Yes	0.829648254	0.748093610	0.722803364
	No	0.833857150	0.756957531	0.731328508
GRU considering both previous and subsequent moments	Yes	0.839500875	0.763506127	0.738497945
	No	0.846575515	0.779274425	0.755235207

Similar results can be obtained even when different values are used as the threshold (see Appendix).

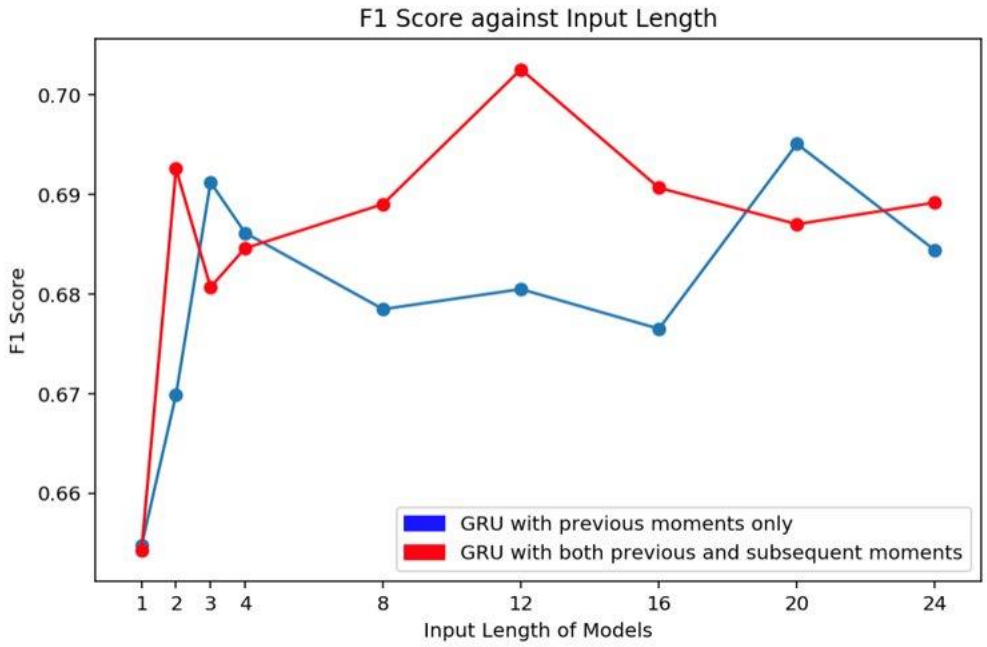
Unfortunately, the results suggest that the chord feature cannot help improve the machine learning component. In both Gated Recurrent Unit models, the metrics drop when the feature is added. One of the possible reasons is that using the chords as binary feature might be insufficient. The inversion of chords might need to be considered as well. Another reason is that the GRU model is already capable of learning the chord relations between the notes, causing the chord feature ineffective.

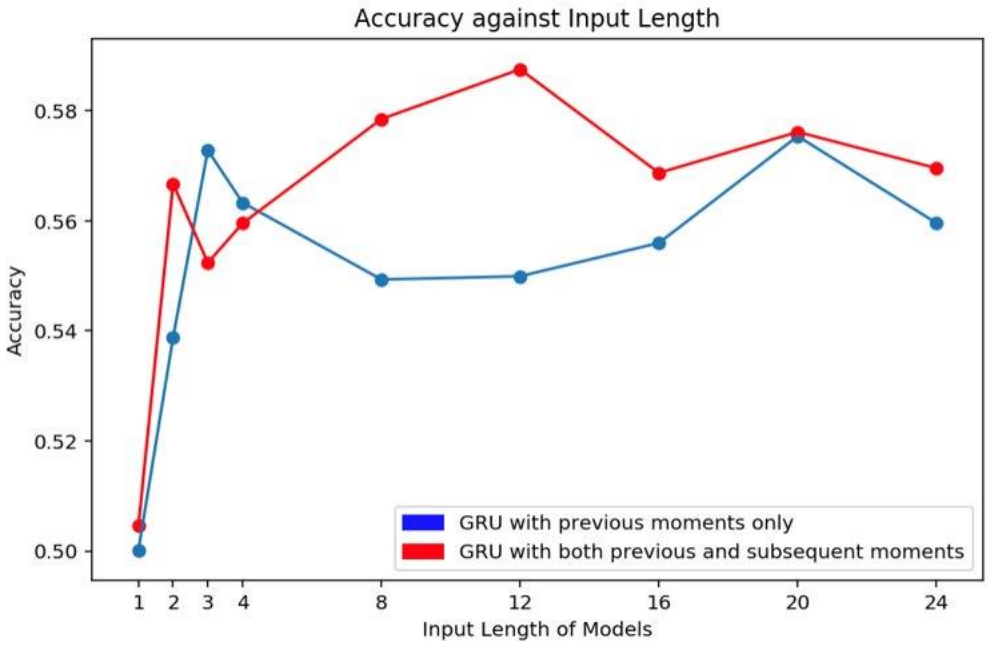
4.6.5.Input Lengths of Models

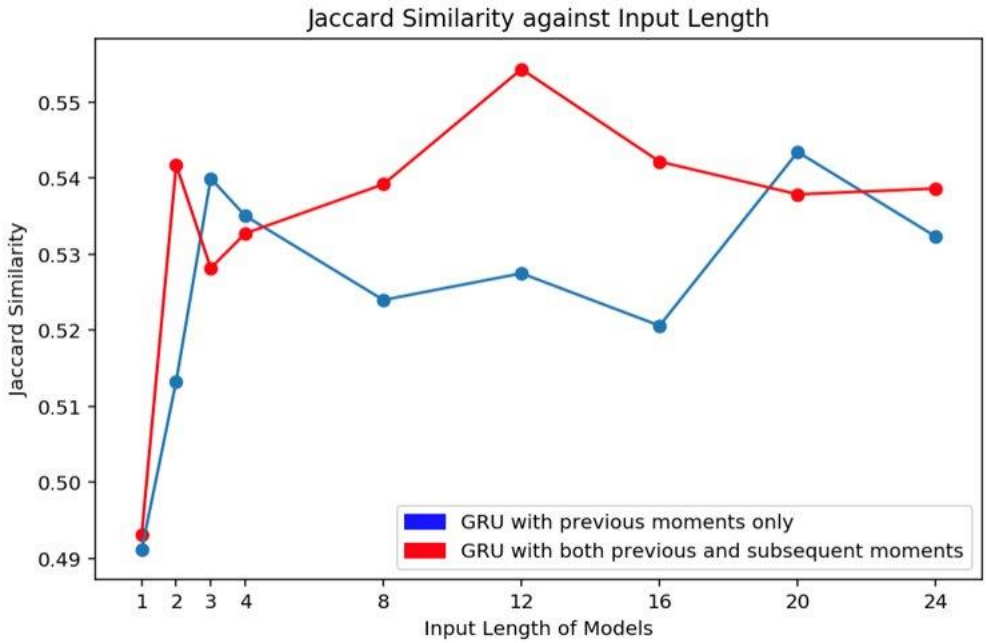
As mentioned previously, when the input length of the Gated Recurrent Unit model changes, different models can be trained, and various reductions can be generated. Therefore, models with different input lengths are evaluated to study the influence and the optimal selection of input length.

Another reduction of Mozart's Cosi fan tutte is used to perform a 5-fold cross validation in the evaluation. The score is split into 5 parts evenly. In the i^{th}-fold run, the i^{th} part of the score is selected as the validation data and the models use the rest of the parts as the training data. The averaged results of the 5 runs are taken as the overall performance of the models.

The following graphs shows the metrics against different input length.







Similar results can be obtained even when different values are used as the threshold (see Appendix).

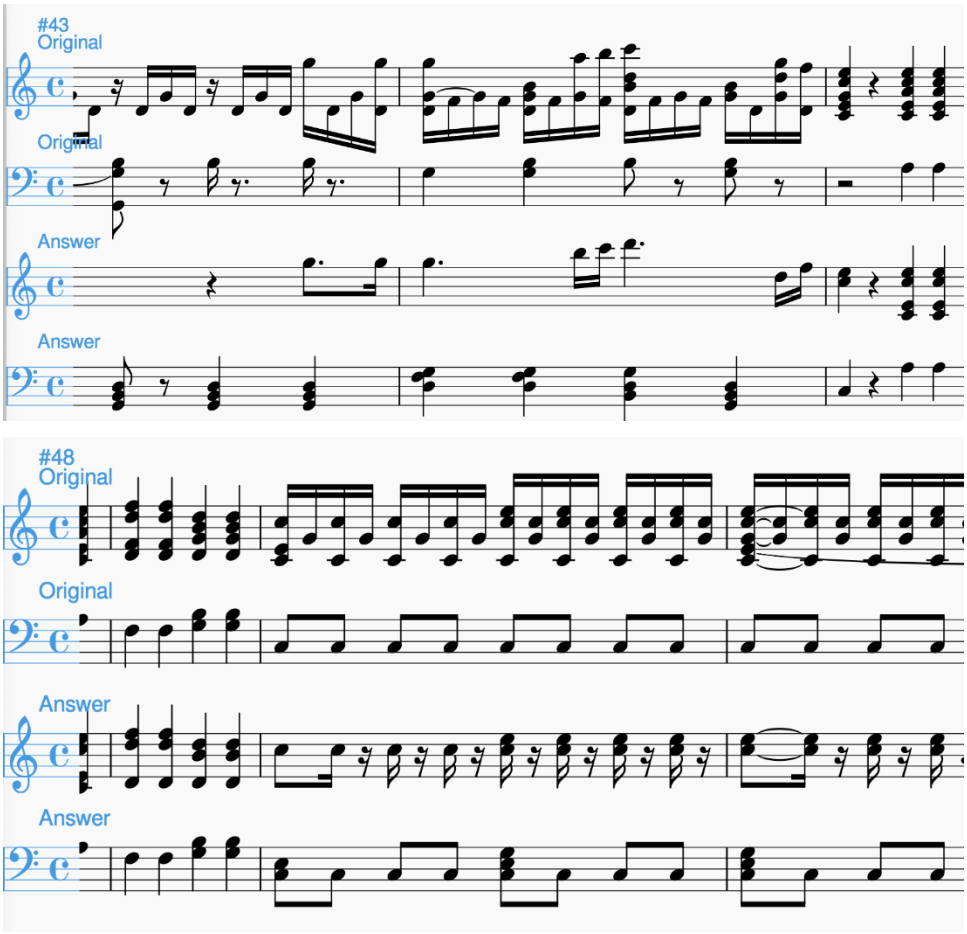
No matter which Gated Recurrent Unit model is used, the results fluctuate when the input length of the model increases. This indicates that the input lengths might have insignificant influence on the performance of the machine learning component. Moreover, the metrics drop drastically when the input length become 1, showing that the adjacent moments give useful information to the model.

4.7. Analysis

4.7.1. Reduction of Cosi fan tutte

A segment of Cosi fan tutte reduction generated by a Gated Recurrent Unit model considering both previous and subsequent moments is used to analyse the performance of the machine learning component.

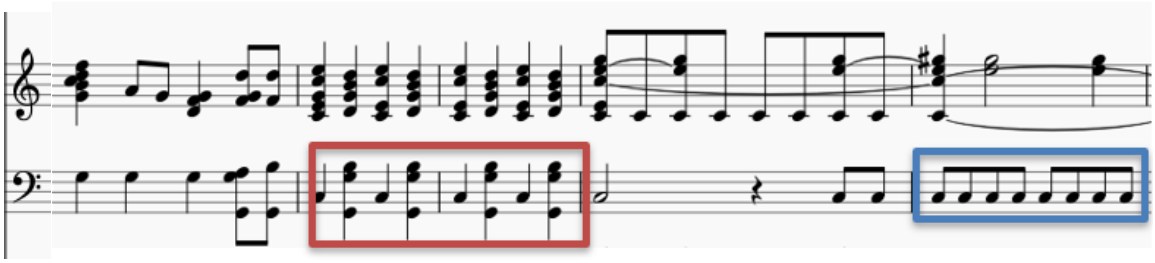
The model is trained using the remaining part of the Answer Score of Cosi fan tutte, which is reduced by Dr. Lucas Wong. In general, the reduction greatly simplifies the original score and is more playable. Many repeating notes are kept and discarded alternatively, and the rapid parts are often removed in the reduction. Since the training data and testing data are from the same reduction, it can be expected that the model should generate a reduced score similar to the answer score.



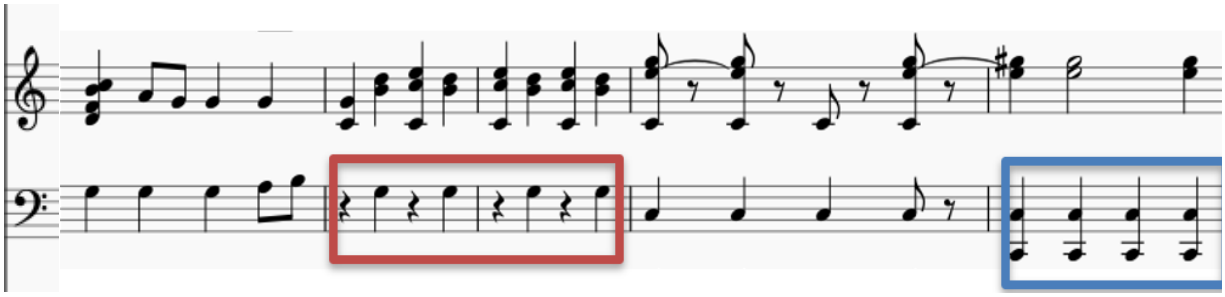
Example of the Reduction used to train the model

4.7.1.1. Reduction Style

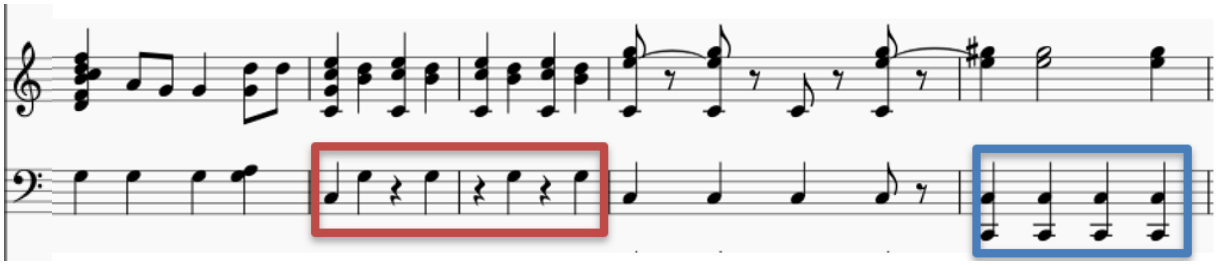
Generally, the model is capable of learning the reduction style, which can be shown from the following measures that most of the notes can be predicted correctly. Specifically, in the last measure of the example, the model successfully learns to reduce the quavers(eighth note) into crotchets(quarter note) in the left-hand part. Also, in second and third measures of the example, the models can learn to reduce the crotchets into rest notes, except for the first beat of the second measures. As we can see, the model can greatly learn the reduction style from the training example.



Original Score



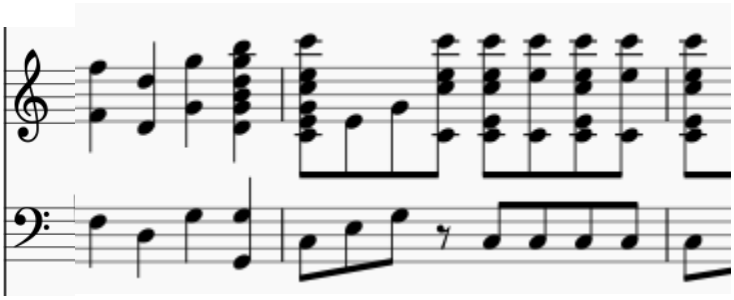
Answer Score



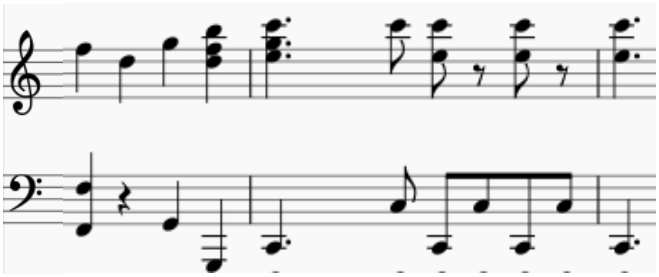
Reduced Score

4.7.1.2. Harmony Preservation

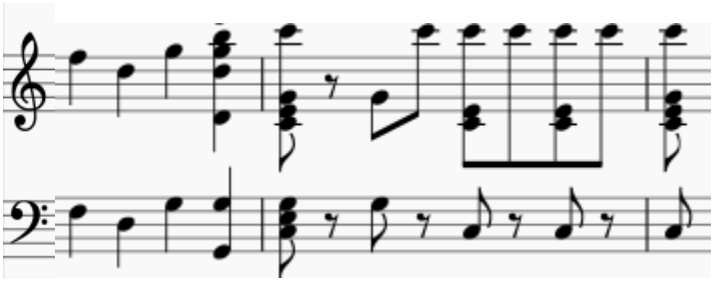
While the model is able to learn the reduction style, the harmony of the piece can also be preserved. Although some of the predictions are incorrect, it can be seen that those incorrect moments had their number of the notes reduced. More importantly, the incorrect predictions, whether there are newly added notes or not, can still match with the chord that the original notes belong to. It indicates that the Gated Recurrent Unit model can learn from the combination of notes from the training data and maintain the harmony of the reduction.



Original Score



Answer Score



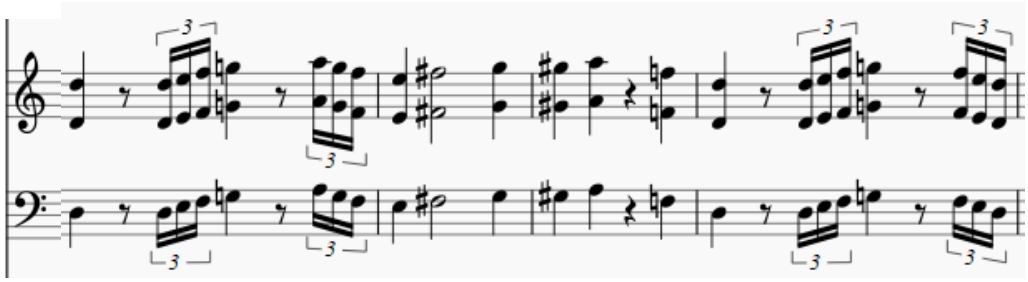
Reduced Score

The above measures can be used as an example. Although the predictions are not completely correct, the model tries to reduce a certain number of notes. In addition, in the first beat of the last measure, a new note G-4 is generated by the model. Even though the generated note is incorrect, it is still within the chord that the others notes

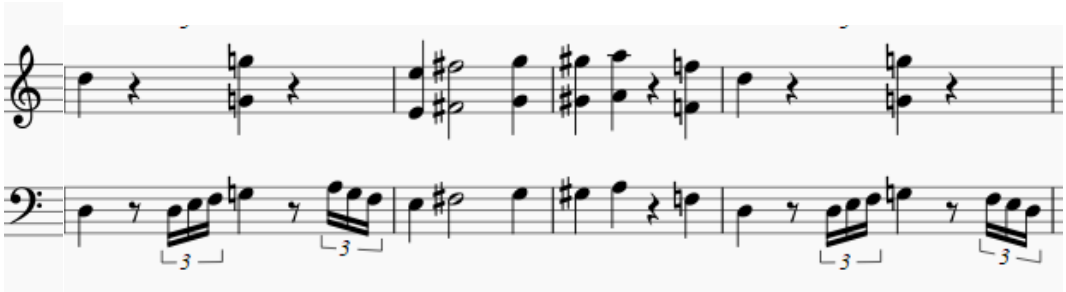
of that onset belong to. As a result, the generated notes and the incorrect predicted did not destroy the harmony.

4.7.1.3. Tempo Overlooking

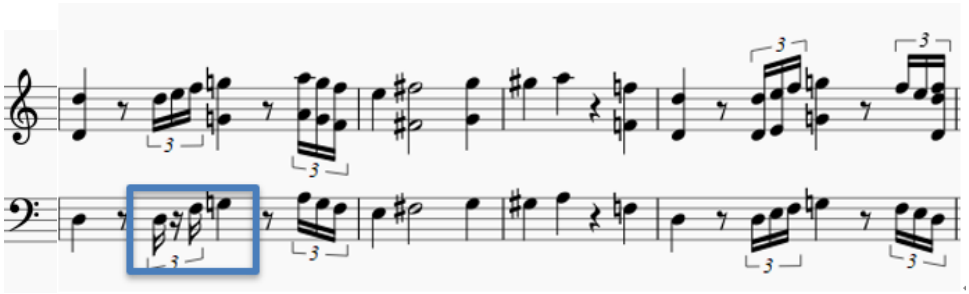
There are still rooms of improvement for the machine learning model. Since the notes are separated according to the moment they belong to only, the tempo of each notes cannot be learnt by the model well. No matter how long the moments last for, they are all treated as the same and each occupies one unit in the sequence. As a result, the Gated Recurrent Unit model is not able to distinguish between a fast melody and a slow melody well, resulting in reducing both melodies in a similar way.



Original Score



Answer Score



Reduced Score

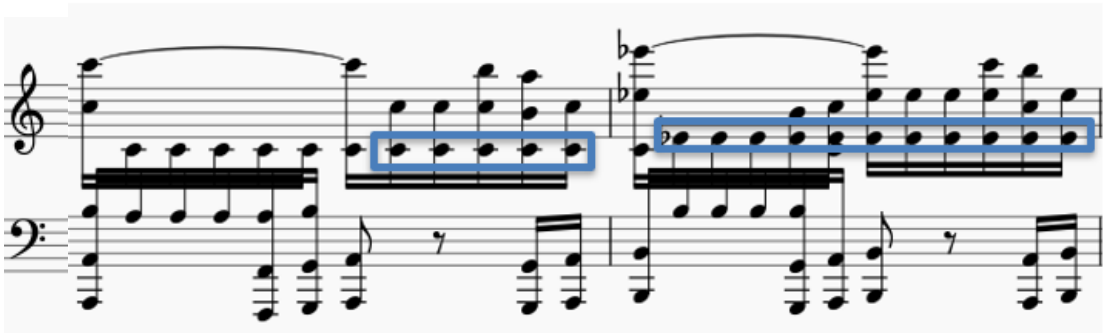
As shown in the measures above, the model mishandles the triplets in the first measure. Neglecting the duration of the notes, it treats the triplets as normal notes

and applies the "reduction style" of keeping and discarding notes alternatively to the triplets, causing a break in the three triplets.

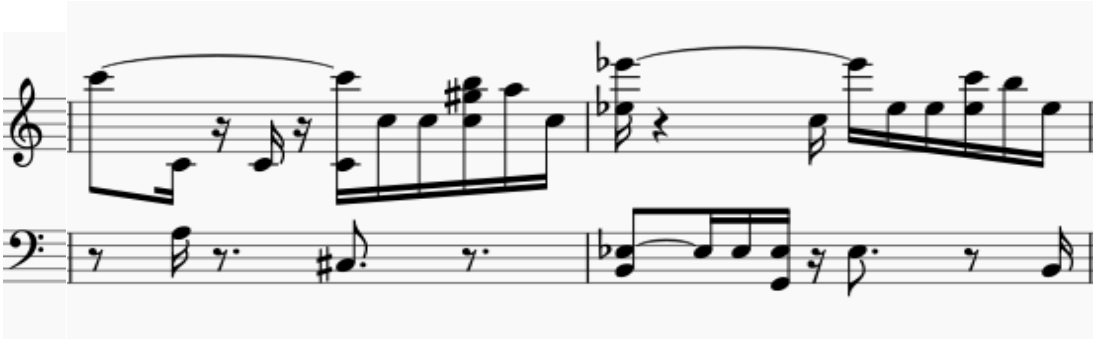
4.7.2.Application on Other Musical Pieces

The same model is used to apply the same reduction on another musical pieces. Obtained from musescore.com, a segment of Beethoven's Symphony No.7 Movement 1 is also reduced by the aforementioned model to analyse how well the machine learning component can reduce the score according to the given style.

The reduction result shows that the model is capable of reduce the score with the given style. The original score contains a large number of notes on most moments, and the model successfully reduce the number of notes. As shown in the following examples, in the treble clef, the repeating notes (C4 in measure 1 and E4 in measure 2) are greatly reduced. In the bass clef, multiple notes are reduced as well, following the simple style of the training data. Even though many notes are removed, the harmony is not destroyed. It is shown that the reduction style can be applied on this musical piece.

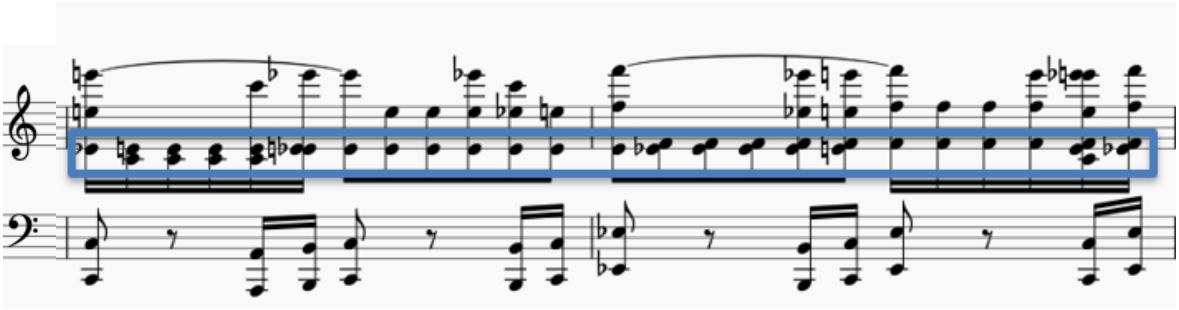


Original Score

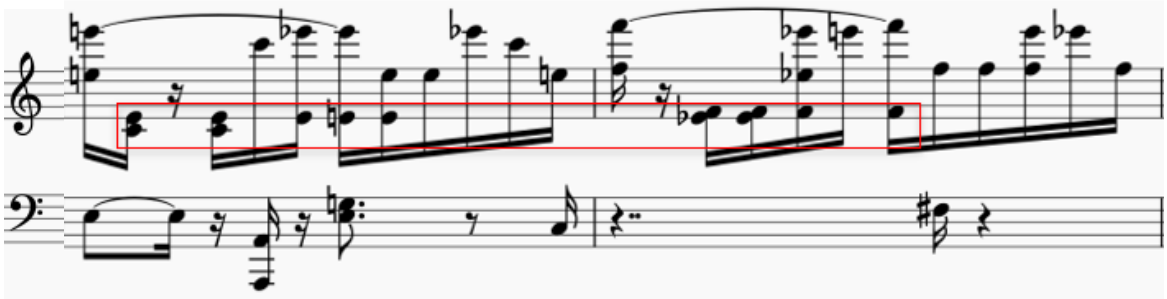


Reduced Score

Despite the success in simplifying the score, inconsistency occurs in the reduced score. It can be seen that for some consecutively repeating notes, a few of them may not be reduced while the rest of them are discarded, failing to apply the same reduction consistently. It could be caused by the different combinations of the set of notes in each moment. Another possible reason is that the previous moment and the subsequent moment of the notes of the input are different so the notes are treated differently.



Original Score



Reduced Score

5. Post-Processor

5.1. Motivation

The machine learning component now outputs a reduced score by using the trained model. However, the reduced score might not be playable all the time. Sometimes, there might be too many notes to be played at once. Sometimes, there might not be enough time for the pianist to move from one position to another to play the notes.

Therefore, the post-processor is responsible to increase the playability of the reduced score, which benefits the pianist.

The post-processor developed by Ng has some room for improvement. Firstly, the program occasionally performs excessive octave transposition. The playability of the score may even decrease. Secondly, the algorithms used do not evaluate how difficult the score is to be played. The results might still be difficult to be played, or sometimes over-simplified which cannot keep the harmonies of the original score. Thirdly, the extent of score simplification is fixed. The processed score may not suit every users' piano skill level. Lastly, the post-processor developed by Yiu sometimes destroys the harmony because it may remove multiple notes at an onset.

Difficult to find an optimal arrangement for the reduced score, we tackled these problems using a stochastic approach to attain sub-optimal solutions. Also, the algorithms are user-configurable, which can generate various versions with different playing difficulties. Moreover, we would like to preserve the harmony of the reduced score.

5.2. Ideas

To evaluate the playability of the piano score, it would be better if we have the information of how the pianist will play the score, which is the finger arrangements. Then, we can calculate base on the finger and hand movements and the range of the hands etc. However, every pianist has their style to play the score. We can only estimate how the pianist will play the notes.

After assigning the finger arrangements to the given notes, we can calculate the "cost" of the arrangement. The score is more playable if the total "cost" of the arrangements of the whole score is lower. Also, we hope that we would modify as fewer notes as possible if the notes can possibly be played. Penalties will be added during the optimization.

5.3. Playability Calculation Model

This model is used to calculate the playability of a score. The score is evaluated moment by moment. The sum of penalties of every moment of the score will be used to determine the playability of the score. A low total penalty denotes that the score is highly playable.

In the model, there are four kinds of penalties. The total penalties will be the sum of all four kinds of penalties.

5.3.1. Finger Busyness Penalty

Sometimes, there may be a moment exists in the reduction score that there are more than 10 notes to be played. This is impossible because a human only has 10 fingers. Moreover, it is difficult for the pianist to play multiple notes for consecutive short moments. Penalties will be given to those moments which have the above situations.

5.3.2. Range Penalty

For the assigned finger arrangements, some of the notes may not be reached even the pianist stretch their hands. Therefore, penalty will be given if the distance between the first note and last note of one hand exceeds the hand size.

5.3.3. Hand Movement Penalty

For a certain pace, it is difficult for the pianist to move their hands vigorously. For the same moments of two scores with the same finger arrangements, the one with higher pace is more difficult to be played. Therefore, penalty will be given to distant hand movement.

5.3.4. Number of Changes Penalty

The main purpose of the post-processor component is to increase the playability of the score while keeping most of the notes in the reduced score. Therefore, excessive changes are discouraged by giving extra penalties for every change.

5.4. Hill Climbing

Hill climbing is a heuristic search algorithm to solve optimization problems. It attempts to find a sub-optimal solution and the solution is improved repeatedly until some condition is obtained. This algorithm may not find the optimal solution, but a sufficiently good solution can be found in limited time.

The algorithm initially assigns fingers to the notes. After that, it will try to reduce the moment in the score that has high penalties by changing notes by octaves (higher/lower) if possible. Since the notes are generated by the machine learning component, we wish to modify as fewer notes as possible.

The optimization algorithm goes as follow:

Repeat until no possible further changes:

Randomly modify the notes and recalculate the new penalty

If the new penalty is lower than the original one then commit the change

Otherwise, revert the change

The users can now select the extent of score simplification based on their piano skill level. They can simply change the input parameters to the model to get the corresponding processed score. Also, the algorithm prevents over-simplification problem.

5.5. Results

Evaluating the performance of the post-processor is difficult because playability is subjective that different pianists may have different views on the playability. Therefore, we tried to compare the results and identify the improvements in the samples.

In this section, there will be some examples of the comparison between reduced score and processed score. "Before" in the examples is the reduced score. "After" in the examples is the processed score.

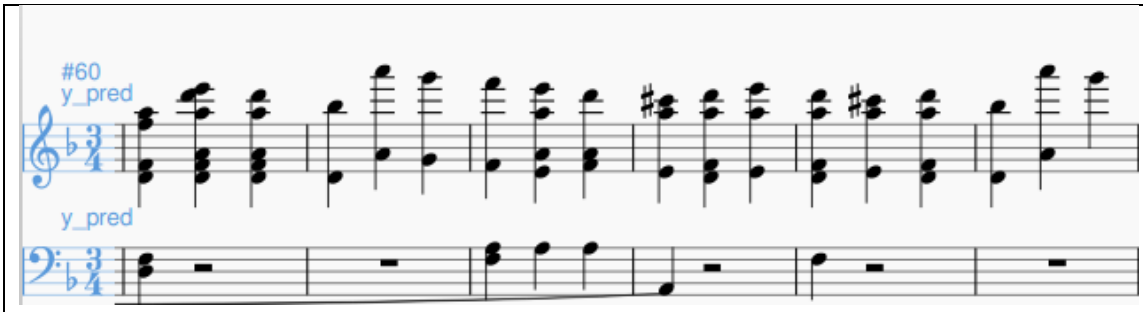
Here, we consider the playability of each moment, which is to consider the difficulty to play all the notes at the same onset.

Several parameters have been adopted to the post-processor to figure out the optimal results. The parameters are the weight to the penalties defined in the playability calculation model. The following table shows the weights used in different trials.

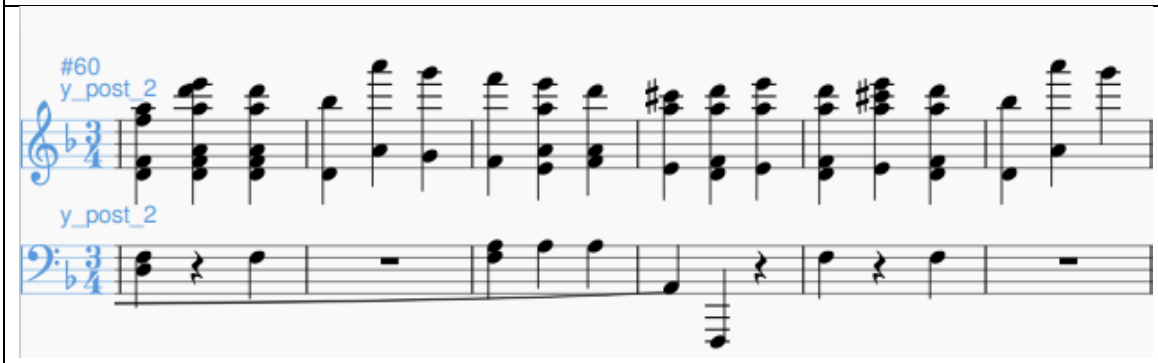
Trial #	hand size	Finger Busyness penalty	Range penalty	hand movement penalty	Number of changes penalty
1	12	1	1	1	1
2	12	10	10	10	1
3	12	10	100	1	10
4	12	1	10	10	10
5	12	1	1	10	1
6	12	1	100	1	10
7	12	100	100	10	1

Beethoven's Symphony No.9 2nd movement is used to tune the parameters of the post-processor. We can see the result of trial #1, with all parameters equal to one, is not satisfying in terms of increasing the playability. This suggests that the parameters need to be tuned in order to have satisfying results.

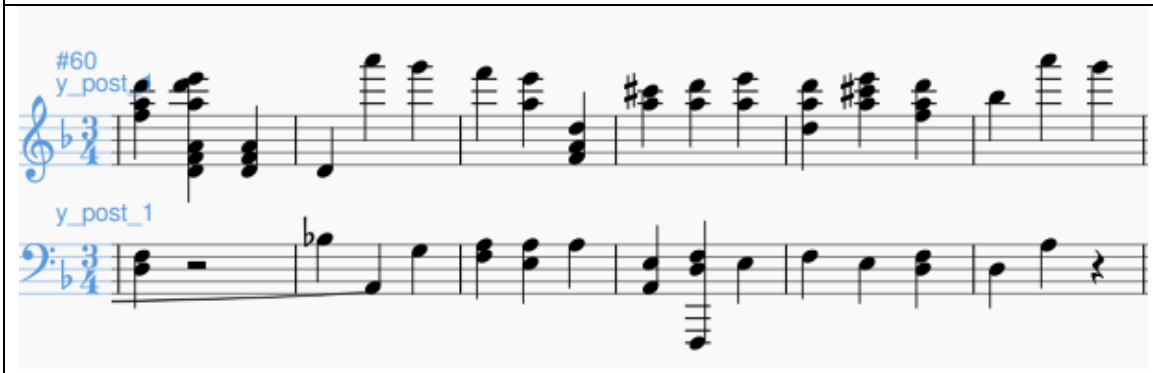
It is found that the model with parameters in trial #3 performs the best among all the trials. It can increase the overall playability of the whole piece while keeping the melody of the musical piece. For examples, in measure 60-65 of the trials we can see that with the parameters in trial #2, trial #4 and trial #7, the overall playability of the musical piece are slightly increased, but overall playability of trial #5 and trial #6 are not increased .



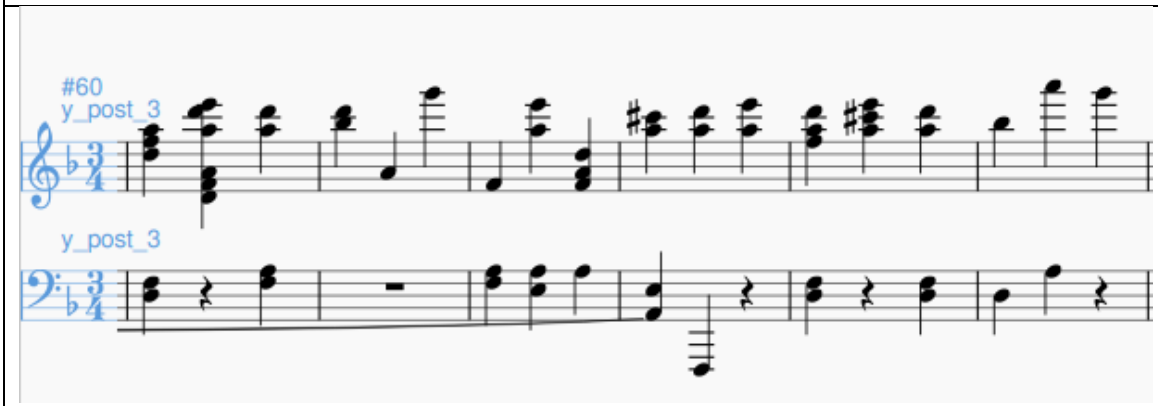
Measure 60-65 of the reduced output of Beethoven's Symphony No.9 2nd movement



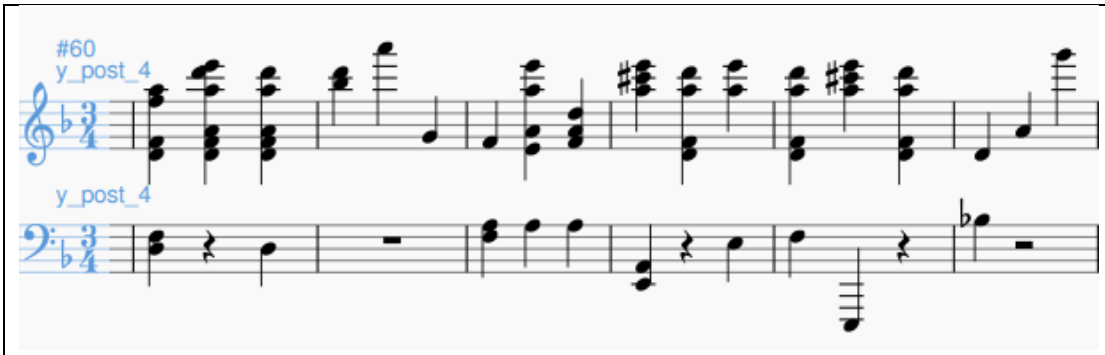
Measure 60-65 of the of processed score (parameters trial #1) of Beethoven's Symphony No.9 2nd movement



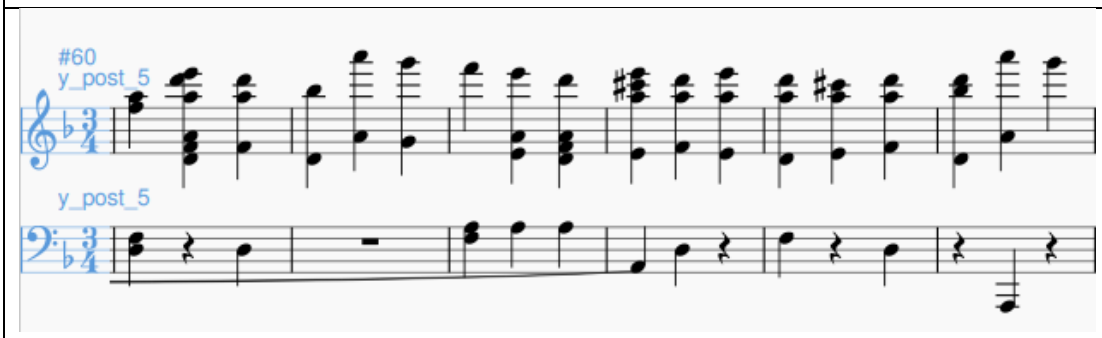
Measure 60-65 of the of processed score (parameters trial #2) of Beethoven's Symphony No.9 2nd movement



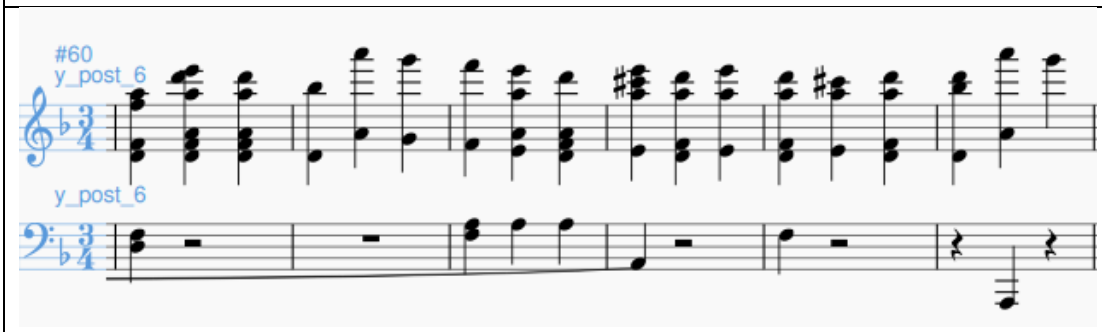
Measure 60-65 of the of processed score (parameters trial #3) of Beethoven's Symphony No.9 2nd movement



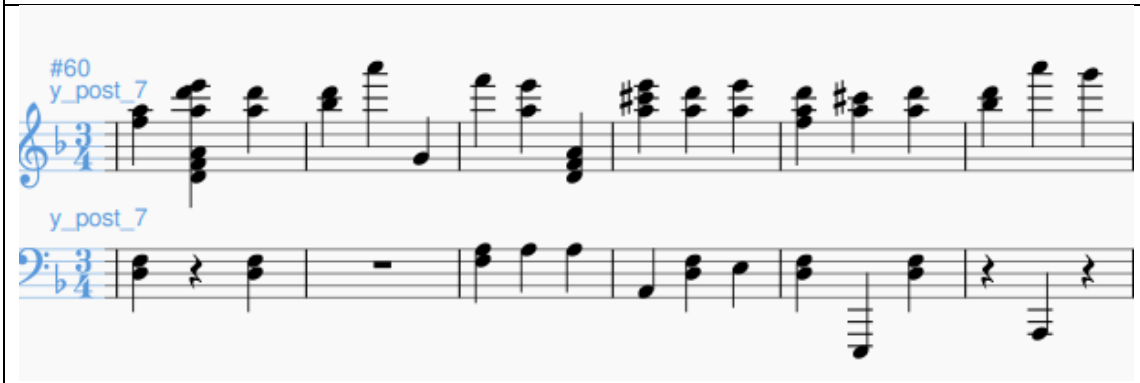
Measure 60-65 of the of processed score (parameters trial #4) of Beethoven's Symphony No.9 2nd movement



Measure 60-65 of the of processed score (parameters trial #5) of Beethoven's Symphony No.9 2nd movement

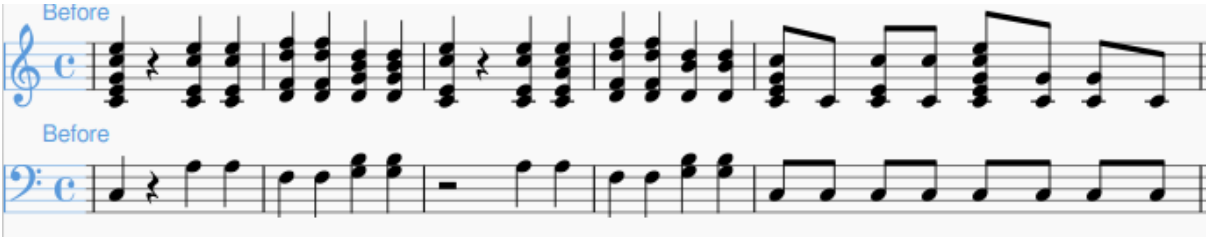


Measure 60-65 of the of processed score (parameters trial #6) of Beethoven's Symphony No.9 2nd movement

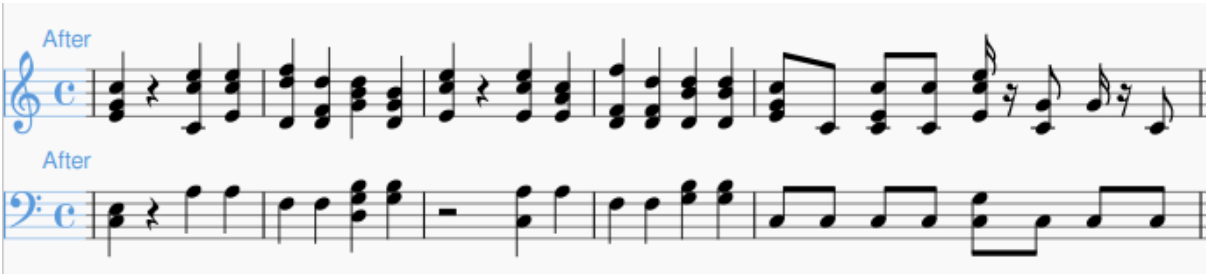


Measure 60-65 of the of processed score (parameters trial #7) of Beethoven's Symphony No.9 2nd movement

Mozart's Cosi fan tutte is also used to evaluate the post-processor with the optimal parameters which are found in the previous part, and it successfully increases the overall playability of the whole piece. For example, in measure 44-48, some of the notes are removed, the score becomes more easier to be played because the number of the notes at the same onset is reduced. Even though some of the notes are removed, the melody can still be kept.



Before: Measure 44-48 of the reduced output of Mozart's Cosi fan tutte



After: Measure 44-48 of the processed output of Mozart's Cosi fan tutte

6. Miscellaneous Work

6.1. Training Data Collection

Harmonic analysis of the Mozart's Cosi fan tutte and Tchaikovsky's Nutcracker March are provided by Dr. Lucas Wong. The analysis includes the chord features of the notes. The notes that are in chords at each moment are marked manually in both scores as training data. The data is used to investigate the usefulness of the chord features.

6.2. Package Completion

The package now includes the functions of all components, including the pre-processor, machine learning component and the post-processor. It allows faster and easier development and maintenance.

7. Future Work

7.1. Collecting Samples from Musicians

As mentioned before, the rules used in the reduction of the answer score vary and are usually unknown to us, making it difficult to evaluate the performance of the machine learning models. However, to achieve a better understanding about the reduction styles, it is encouraged for us to invite musicians to produce more training samples. As a result, we can understand the rules that the musicians used to perform the reduction and better evaluations and analysis can be performed

7.2. Further Investigations on Chord Features

Due to the limited amount of time, further investigation on the usefulness of the chord features cannot be done. Since it is very time consuming and error-prone to mark the chord features notes by notes by human, only two different scores are used in the evaluation.

Currently, we built a binary feature for each note by considering whether the note is in chord or not. However, the evaluation suggests that the binary feature might not be useful in improve the piano reduction. A possible reason for the unsatisfactory performance might be the misuse of the chord feature. Alternative methods should be adopted in the future to provide more useful information to the machine learning models. For instance, different values can be given to the bass note and root note. Inversions can be considered as features as well.

7.3. Model Ensembling

The aforementioned model stacking cannot perform well in practice. The final classifier requires extra data to train the parameters, making the GRU models perform poorly. In the future, more model ensembling techniques can be experimented to improve the accuracy of the original machine learning models. For instance, different models trained by different parts of training data can be combined to form a larger model as well.

8. Appendix

8.1. Evaluation of Chord Feature using Different Threshold

The aforementioned results used 0.1 as the threshold for the reduction. That is, the notes having a likelihood output by the machine learning model larger than 0.1 will be added into the reduced score. The tables below show the results when other values are used as the threshold.

Threshold used: 0.15				
Testing Data: Mozart's Cosi fan tutte				
Model	Using Chord as Feature?	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Yes	0.645991086	0.582957590	0.484239539
	No	0.647846273	0.579164242	0.486050524
GRU considering both previous and subsequent moments	Yes	0.655657984	0.593652424	0.495444121
	No	0.664077304	0.592258561	0.506076474
Testing Data: Tchaikovsky's Nutcracker March				
Model	Using Chord as Feature?	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Yes	0.846206156	0.774342659	0.746739999
	No	0.847092349	0.777981032	0.750178961
GRU considering both previous and subsequent moments	Yes	0.851432065	0.784328226	0.757057850
	No	0.856227544	0.794444113	0.768121623

Threshold used: 0.2				
Testing Data: Mozart's Cosi fan tutte				
Model	Using Chord as Feature?	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Yes	0.650335226	0.600459378	0.489283769
	No	0.651784226	0.597140531	0.491257349

GRU considering both previous and subsequent moments	Yes	0.661174777	0.611092176	0.502290047
	No	0.668647731	0.608124527	0.510644960
Testing Data: Tchaikovsky's Nutcracker March				
Model	Using Chord as Feature?	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Yes	0.850459220	0.783495612	0.753789729
	No	0.852705488	0.788144294	0.757478560
GRU considering both previous and subsequent moments	Yes	0.854870863	0.791446372	0.762052660
	No	0.860211388	0.801235483	0.772653889

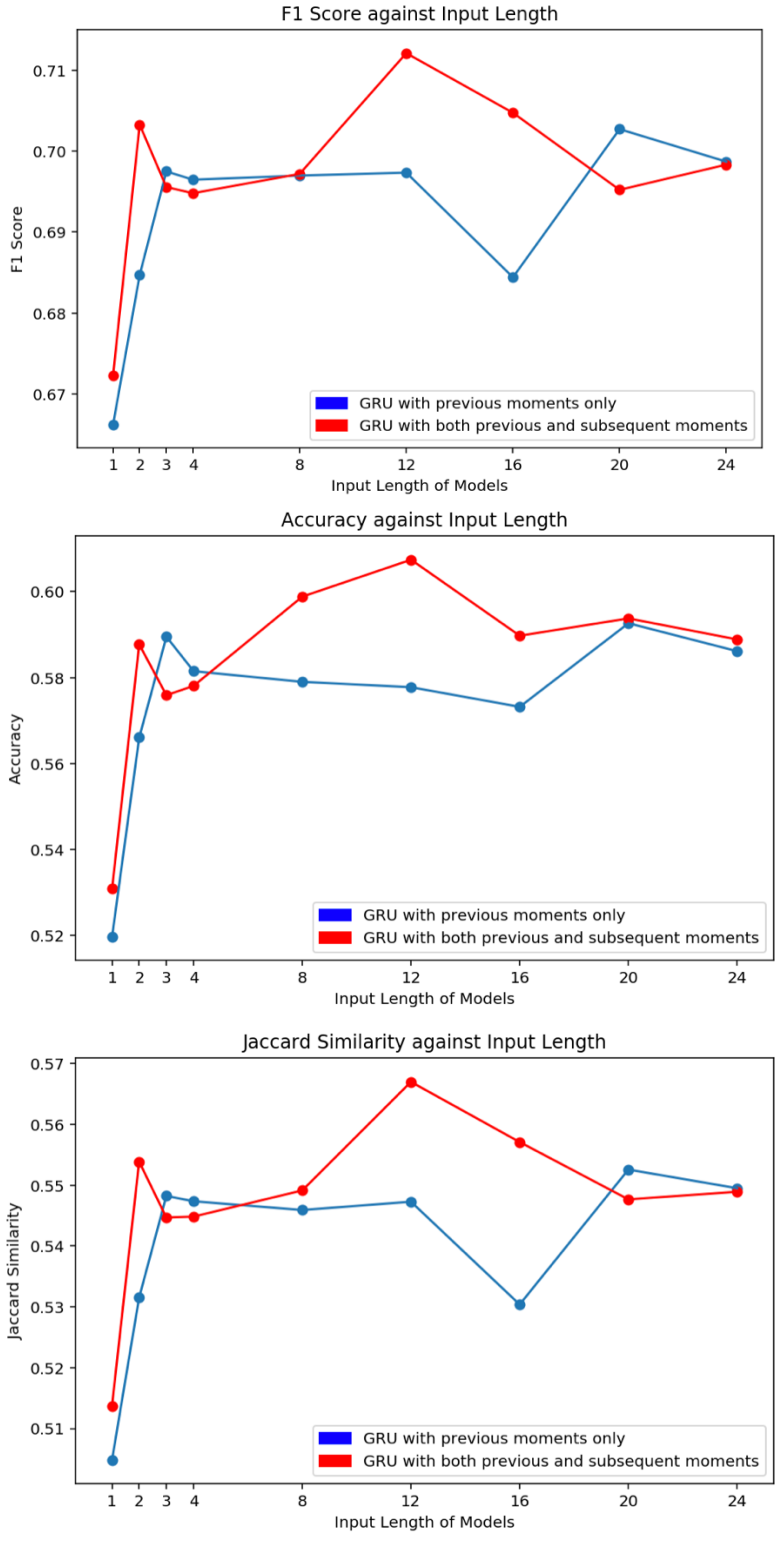
Threshold used: 0.25				
Testing Data: Mozart's Cosi fan tutte				
Model	Using Chord as Feature?	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Yes	0.648070319	0.609971566	0.487485153
	No	0.649278755	0.607361401	0.488716931
GRU considering both previous and subsequent moments	Yes	0.661148776	0.621791843	0.503144212
	No	0.664534601	0.615161766	0.506312796
Testing Data: Tchaikovsky's Nutcracker March				
Model	Using Chord as Feature?	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Yes	0.853611949	0.790265502	0.758371379
	No	0.853158892	0.791055798	0.757952254
GRU considering both previous and subsequent moments	Yes	0.857217274	0.796246496	0.765607484
	No	0.864180554	0.808480377	0.778763423

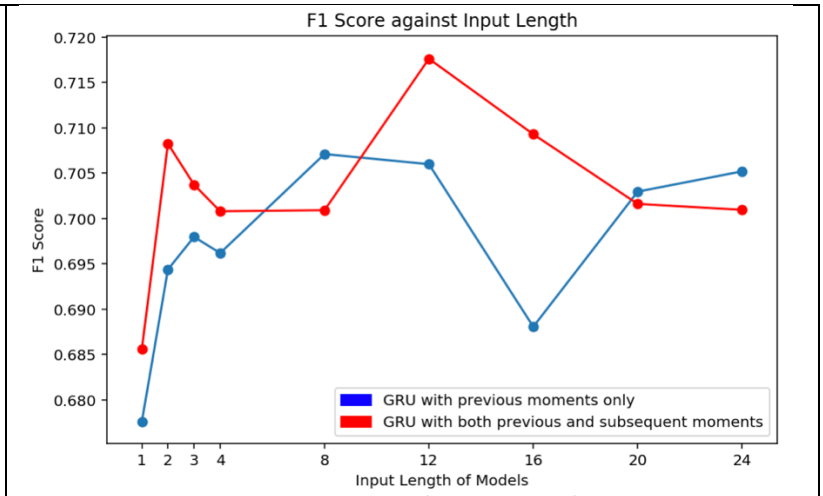
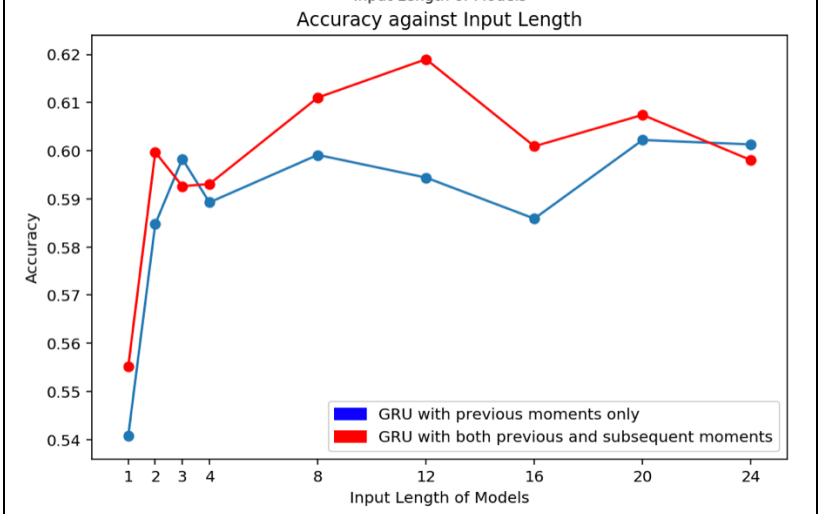
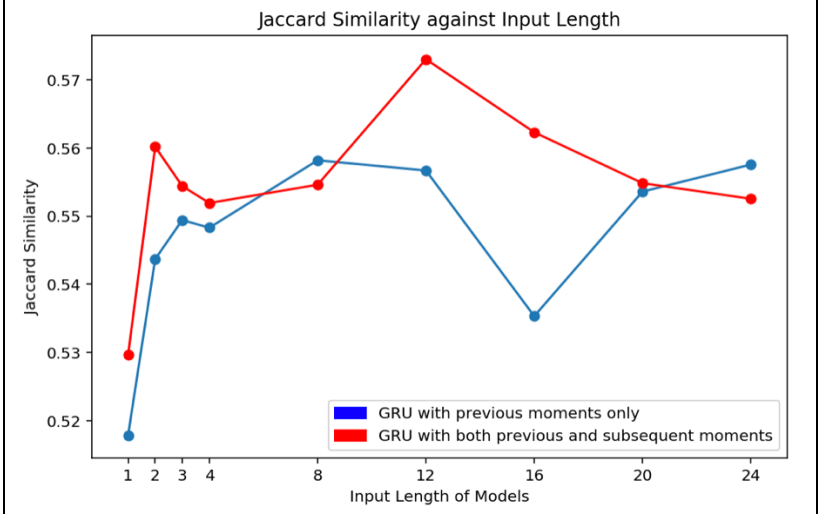
Threshold used: 0.3				
Testing Data: Mozart's Cosi fan tutte				
Model	Using Chord as Feature?	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Yes	0.648456105	0.619255428	0.488095513
	No	0.646535313	0.616299945	0.486511813
GRU considering both previous and subsequent moments	Yes	0.657006831	0.625648002	0.499676341
	No	0.659573695	0.620102279	0.501772617
Testing Data: Tchaikovsky's Nutcracker March				
Model	Using Chord as Feature?	F1 Score	Accuracy	Jaccard Similarity
GRU considering previous moments only	Yes	0.855899476	0.794716960	0.760908747
	No	0.853031740	0.792384404	0.757474559
GRU considering both previous and subsequent moments	Yes	0.858495259	0.799108261	0.767281559
	No	0.866976779	0.813950144	0.782224706

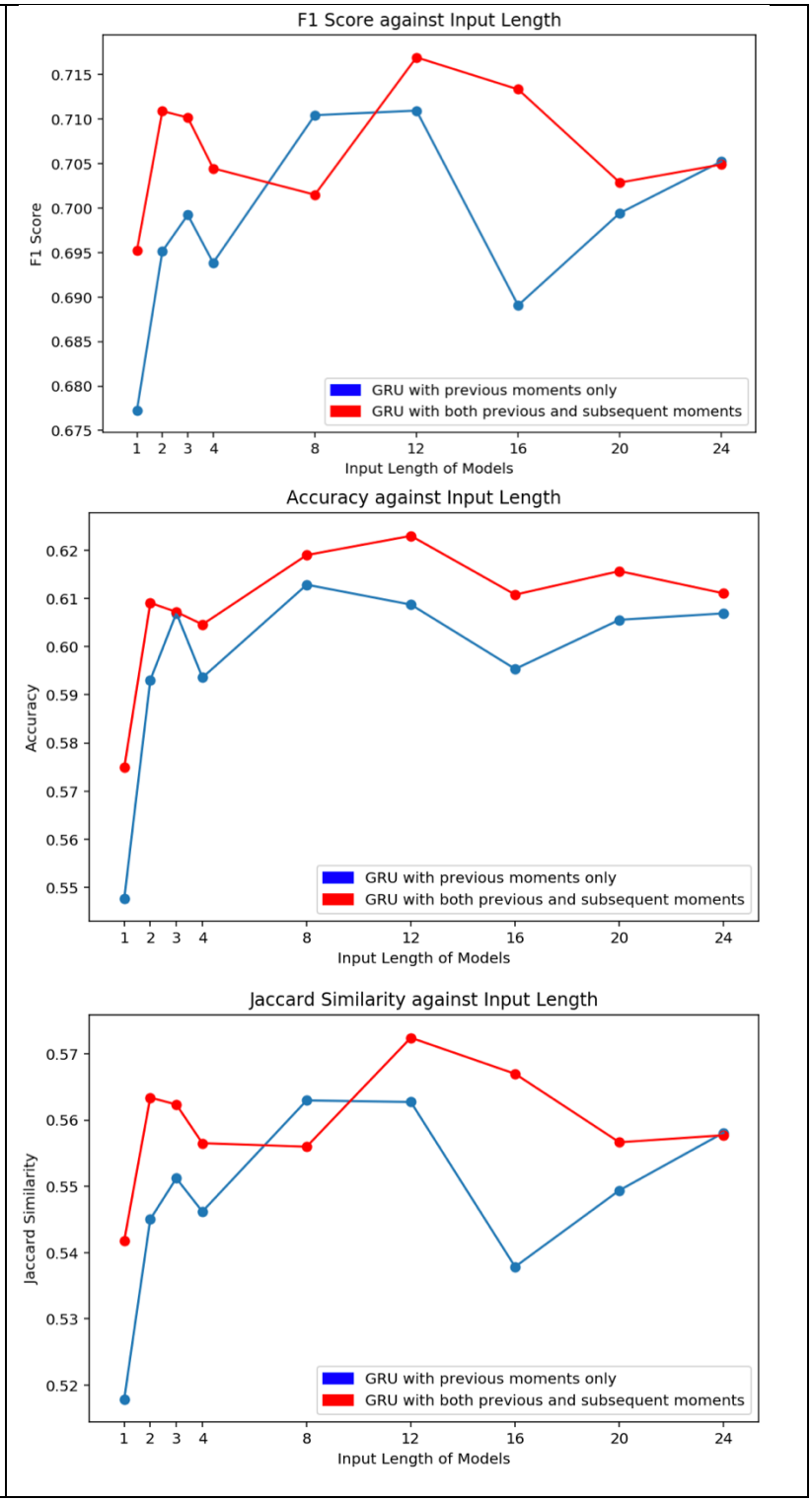
These results are similar to the aforementioned evaluation that the chord feature is not effective on improving the machine learning component. In most cases the F1 score and accuracy become lower when chord is considered by the GRU models.

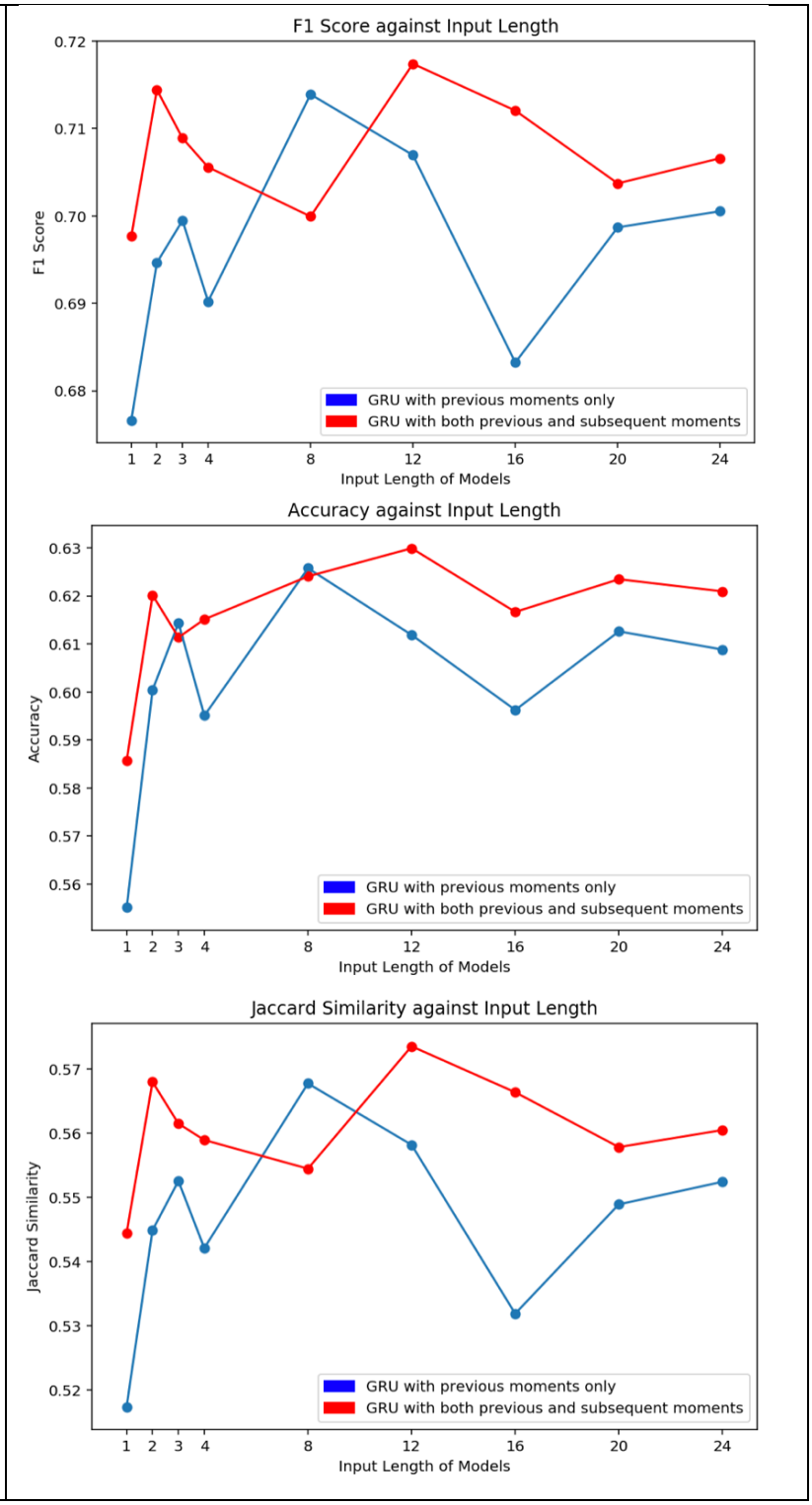
8.2. Evaluation of Input Lengths using Different Threshold

Similar to the above, the following graphs show the results when values other than 0.1 are used as the threshold of the machine learning component to add the note.

Threshold used	Metrics against input lengths
0.15	

0.2	 F1 Score against Input Length F1 Score Input Length of Models GRU with previous moments only GRU with both previous and subsequent moments  Accuracy against Input Length Accuracy Input Length of Models GRU with previous moments only GRU with both previous and subsequent moments  Jaccard Similarity against Input Length Jaccard Similarity Input Length of Models GRU with previous moments only GRU with both previous and subsequent moments

0.25	 F1 Score against Input Length F1 Score Input Length of Models GRU with previous moments only GRU with both previous and subsequent moments Accuracy against Input Length Accuracy Input Length of Models GRU with previous moments only GRU with both previous and subsequent moments Jaccard Similarity against Input Length Jaccard Similarity Input Length of Models GRU with previous moments only GRU with both previous and subsequent moments

0.3	

All results fluctuate when the input length of the GRU models increases. Nevertheless, the scores when input length of 1 is used are still the lowest.

9. References

[1] Charles Sutton and Andrew McCallum, in *An Introduction to Conditional Random Fields*, Foundation and Trends in Machine Learning 4.4, 2012, pp. 267-373.

[2] Stuart Russell and Peter Norvig, in *Artificial Intelligence – A Modern Approach*, New Jersey, Prentice Hall, 2012, pp. 10-11, 727-736.

[3] Schmidhuber Jürgen, and Sepp Hochreiter, “Long Short-Term Memory,” *Neural Comput,* vol. 9, no. 8, pp. 1735-1780, 1997.

[4] C. Olah, “Understanding LSTM Networks,” 27 8 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed 27 11 2018].

[5] Chung Junyoung, et al. , “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” 2014.

[6] R. Gandhi, “Introduction to Sequence Models—RNN, Bidirectional RNN, LSTM, GRU,” 27 6 2018. [Online]. Available: https://towardsdatascience.com/introduction-to-sequence-models-rnn-bidirectional-rnn-lstm-gru-73927ec9df15. [Accessed 27 11 2018].

[7] Yiu, Choi, "Automatic Piano Reduction (Backend II): Algorithms", 2017.