Time Period Classification for Piano Music Using Sequence Modeling

Shivang Singh
Department of Computer Science
University of Texas at Austin
shivang.singh@utexas.edu

Mengning Geng
Department of Computer Science
University of Texas at Austin
mengnninggeng@utexas.edu

Abstract

This paper explores the use of sequence modeling to do time period classification for classical music. We explore the use of a long short-term memory (LSTM) network to model music. We do this in two steps. (1) In the first step, we design an upstream task that takes in input sequences of music notes. (2) Secondly, we use this model to classify music sequences into musical time periods. We derive this set of mappings through finding lists of time periods and the respective pieces that correspond to the period. We show that by first training a language model on a set of unlabeled data and then fine tuning on a smaller set of labeled data, we are able to improve performance. Through this modeling, we hope to learn what factors and features are the most useful in delineating between time periods. Furthermore, we would like to explore the model that we build through probing to learn more about musical time periods.

1 Introduction

Time periods in music have distinct traits that define the difference between that time period and other time periods. This paper explores a way to classify sections of music as belonging to certain time period. Furthermore, it seeks to uncover the factors that have the most influence on such a classification. Previous works in this field have explored features that can be extracted from the piece of music through feature engineering and have used this set of features in different classifiers. (Kempfert and Wong, 2018; Sadeghian et al., 2017) Other works in this area use minimally preprocessed data to feed into convolutional networks, which are able to learn features required for classification.(Verma and Thickstun, 2019) These models are able to leverage convolutional layers to learn important characteristics about the musical composition segments. Furthermore, the classification task that they are interested with is composer identification. Some other downstream tasks that have been explored are genre classification and mood classification. (Defferrard et al., 2018). While these approaches have been powerful, they require a large amount of labeled data, which is not always available for all classification tasks.

In this paper, we introduce two new ideas. First, we define a different classification task, which we believe is important to understanding the evolution of music. This classification task is that of classifying sequences of music notes to a bucket corresponding to the time period where they belong. This classification task has unique challenges that composer classification does not. While composers are known to have and still reuse chord progressions within their works, it is relatively subjective as to what the exact changes are between time periods. Second, we use the idea of leveraging language modeling to model music data for classification. In this task, we train a model to understand music in an upstream task. This model learns to predict sequences of notes and their relative duration. The purpose of the upstream task is to learn inherent characteristics about music for the model to better adapt to the task of classification. We then use this upstream model for the downstream task of time period

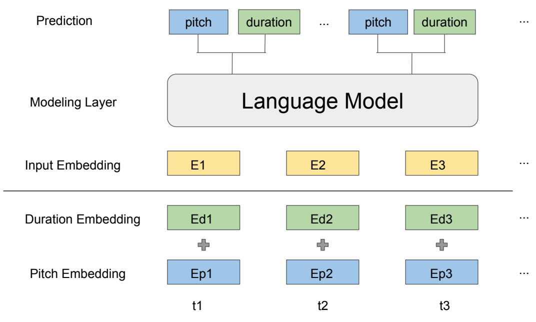

Figure 1: The overview of our music generation model. As opposed to a normal language model, we have two inputs, namely, pitch and duration for each note. The two inputs are first converted to an embedding representation and added together to get the real input for the language model. For each time step t, the output is also a pitch and duration of a note.

classification. By using this methodology, we hope to both improve the performance of classification and also make our method as well as our model more tractable for other applications.

We model a sequence of notes using the standard LSTM (Hochreiter and Schmidhuber, 1997), with the input composed by the pitch and duration of a note. For language modeling, we also predict the pitch and duration for a note at each time step. We use the GiantMIDI dataset (Kong et al., 2020) for both training and evaluation. By doing the language modeling over music notes, we find that there does exist long-term dependency in music generation which indicates that the composer may actually write the music pieces as a whole. Additionally, we find that by understanding these long term dependencies across music compositions, we can perform better in classifying these pieces. We are able to develop an understanding that allows us to better categorize these music pieces.

2 Dataset

The dataset contains over 10,000 compositions spanning from a wide variety of piano composers. The MIDI file format contains a detailed illustration of the inner structure of compositions. By leveraging the vast amount of data that is stored (such as data on pitches, duration, and start times) of different notes, we can accurately model the structure of music. Furthermore, the dataset specializes in piano music. The composers in the dataset wrote these pieces to be played in a piano solo format. Thus our problem focuses on that instrumentation. The statistics of this dataset is shown in Table 1. We randomly shuffle the whole datasets and take 90% of the data as our training set and the rest as evaluation set.

Dataset	Composers	pieces	Hours
piano-midi.de	26	571	36.7
Classical archives	133	856	46.3
Kunstderfuge	598	-	-
MAESTRO	62	529	84.3
MAPS	-	270	18.6
GiantMIDI-Piano	**2,786**	**10,854**	**1,237**

Table 1: A comparison of different Piano MIDI datasets. Here, we see that the GiantMIDI-Piano has much more pieces of music than any other datasets published before.

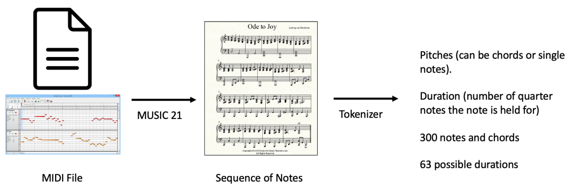
MUSIC 21
Ode to Joy
Tokenizer
Pitches (can be chords or single notes).
Duration (number of quarter notes the note is held for)
300 notes and chords
63 possible durations
MIDI File
Sequence of Notes

Figure 2: This is the overview of the preprocessing procedure. MIDI files, which are in the format of piano rolls, are converted to sequences of notes. These notes have properties such as pitch and duration. The pitch in piano can consist of a single note and as a series of notes played at the same time (chord). Duration of a note is represented in the number of quarter notes it consists of. Each possible duration in sheet music was encoded as an embedding.

3 Methodology

3.1 Preprocessing

Given a MIDI file, the preprocessing task was to convert the MIDI time series data into a format that is sequential and can be modeled. In order to do so, we leveraged the MIT Music21 library.(Cuthbert and Ariza, 2010) This library provided us with access to a stream of MIDI data. This stream consisted of note object, from which we extracted important information such as the pitch and the duration of each individual note.

3.2 Modeling Music

One piece of music can be thought of as a sequence of notes, where each note is composed by its pitch and duration. Formally, it could be written as a sequence of tuples $(p_1, d_1), (p_2, d_2), ..., (p_N, d_N)$ where N is the length of the sequence. We explore two tasks in this project, namely music generation and music time period classification. We discuss the details of the model we use for each task in the following of this section.

3.3 RNN based Music Generation

The music generation could be modeled using a language model. Given a sequence of tuples, our goal is to maximize the log likelihood of the tuple at time step t given its previous sequence:

$$\text{argmax}_\theta P((p_t^*, d_t^*)|(p_1, d_1), ..., (p_{t-1}, d_{t-1}); \theta), \quad (1)$$

where θ denotes the model parameters and $*$ denotes the gold label. We model the sequence using a LSTM in this work. LSTM processes it by incrementally adding up new content into a single slot of maintained memory, with gates controlling the extent to which new content should be memorized, old content should be erased and current content should be exposed. At time step t, the memory c_t and the hidden state h_t are updated with the following equations:

$$\text{LSTM}: h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow h_t^l, c_t^l \quad (2)$$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \quad (3)$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g \quad (4)$$

$$h_t^l = o \odot \tanh(c_t^l) \quad (5)$$

Since LSTM only takes one embedding as input, we therefore form two embedding matrix $E_p \in \mathcal{R}^{|V_p| \times D}$ and $E_d \in \mathcal{R}^{|V_d| \times D}$ for note pitch and duration respectively. The input embedding of LSTM is computed as $E_p(p_t) + E_d(d_t)$. We take the hidden state at time step t and pass it to a fully connected feedforward layer followed by a log softmax to compute the log likelihood of the gold label. The overview of our model is shown in Figure 2.

To train the language model, we chunk the full music to several pieces using a sliding window with window size l and stride s since a full music contains too many notes which are hard to process. We use perplexity and per-step accuracy as the evaluation metric for our music language model.

3.4 RNN based Music Time Period Classification

The time periods for our model were determined by examining the date when the composition was written and interpolating what time period it belonged to. While this is an imperfect measure, for the scope of the current project, the dates were bucketed into 8 categories. The architecture of the music time period classifier is based on the language model that we have introduced above. Using transfer learning, we sought to use the insights the model gained from the earlier part for doing the classification task. Instead of predicting and generating the sequence that occurs after the current sequence, we use the hidden state to predict the time period classification. After training the sequence model from the earlier part, we froze the weights of the network. A linear layer of the size (hidden layer size, number of classes) was added. We used the labeled data to then train this model. This labeled data was roughly 20 times smaller than the data used for the language modeling task.

4 Experiments

4.1 Language Model Pre-training

We started off by pretraining the language model for the upstream prediction task described in the methodology. For the language model pretraining, we adapt the standard LSTM with only one hidden layer, the hidden size is set to 256 and dropout rate set to 0.1. Both the pitch and du-

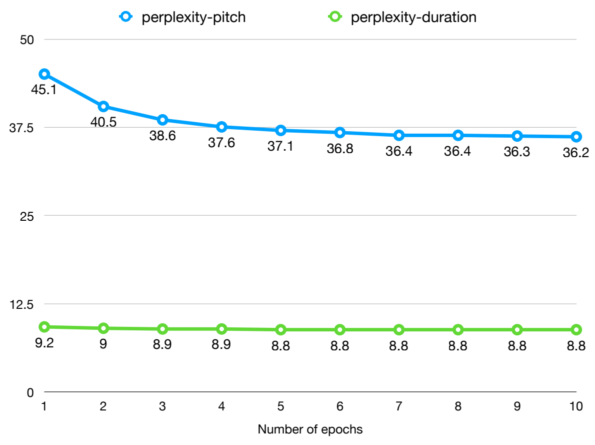

Figure 3: This figure describes the training procedure of the language model. The perplexity of the pitches and the preplexity of the duration refers to the ability of the model in predicting the next pitch or duration respectively.

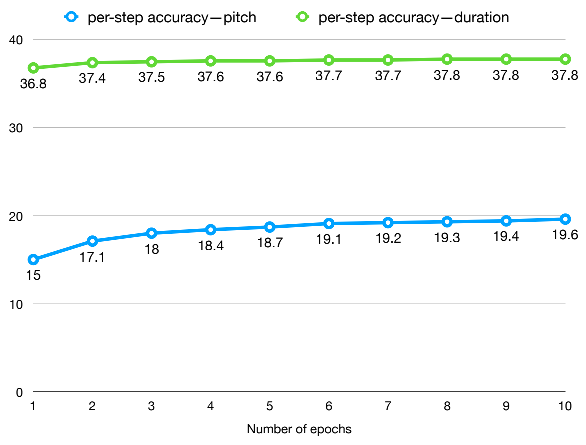

Figure 4: This figure is companied with Figure 3. The per-step accuracy is a more comprehensive evaluation metric that refers to the ability of the model in predicting the next pitch or duration respectively.

ration embedding is initialized using a uniform distribution with range -0.1 to 0.1, the embedding size is also set to 256. We use the Adam optimizer with a initial learning rate set to 1e-3. We use perplexity and per-step accuracy as the evaluation metric for the language model (Irie et al., 2019). It is a measure of how well a probability distribution predicts a sample. It is defined as the following expression:

$$2^{\sum_x p(x) log_2(p(x))} \tag{6}$$

4.2 Results and Analysis

Evaluation at each training epoch Figure 3 and Figure 4 shows how the perplexity and the accuracy change as the training goes. We have several observations here: The perplexity for pitch prediction decreases from 45.1 to 36.2 (a relative 20% decrease), while the number for duration prediction is only 9.2 to 8.8 (a relative 4% decrease). This demonstrate that the duration prediction is much easier than the pitch prediction – a few epochs are enough for the model to converge. We argue it is because the vocabulary size difference that leads to the observation. We see the vocabulary size for pitch and duration is 1869 and 136 respectively. Therefore, predicting a pitch is a harder task than predicting a duration. In general, we see the performance is not as good as a traditional language model trained on natural language. Here are some factors that might affect the performance: (1) Although the Giant-midi dataset already contains much more pieces of music than the previous datasets, it is still much less than the natural language we can easily access from the web. (2) The LSTM is a relatively weak model compared to the trending giant transformers (Vaswani et al., 2017). We plan to explore more about different encoders in the future.

Long-term dependency does exist in music composition! One important factor in the pretraining framework is the window size. Intuitively, the music note generation should not have very long-term dependency (e.g. the start of a song should not affect the middle of it). Is this true for the language model? We conduct an experiment using different window size and corresponding stride size. The results are shown in Table 2. Surprisingly, as we increase the window size from 64 to 512, the model achieves 1% absolute improvement (5% relative improvement) over the per-step accuracy and 2.8 (7% relative improvement) over perplexity on pitch prediction. It shows that there does exist some really long-term dependency when the composers wrote the music

		Pitch		Duration	
window size	stride size	perplexity	per-step accuracy	perplexity	per-step accuracy
64	16	39.2	18.6	9.1	37.1
128	32	37.8	19.0	9.0	37.4
256	64	36.9	19.3	8.9	37.7
512	128	36.4	19.6	8.8	37.8

Table 2: The language modeling performance for pitch and duration with respect to different window size and stride size. For perplexity, the lower, the better. For per-step accuracy, the higher the better. We see that larger window size consistently leads to better performance.

although they may note be aware of it themselves.

4.3 Down-stream Music Time Period Classification

Models over Time Period	Accuracy	F1-score
Baseline	34.86	37.41
Pretrained Head	37.73	41.27

Table 3: These are the result of running the classification model. The baseline model, is a model that is trained from scratch. The second model is the model that the paper has outlined. It uses the pretrained head from the language modeling section and trains a linear layer upon it. Also, it is important to not that both model outperform guessing and predicting the majority class (which in this case is the class with roughly 25 percent of the distribution).

In the downstream task, we took the model that was trained in the upstream analysis and copied the embedding layers as well as the LSTM module into our downstream model. The model was then trained using a 1e-4 learning rate with Adam optimizer. The sizes of the embedding layer and the hidden layer of the LSTM were kept the same. A new linear layer was added to map the output of the LSTM model to classes. The model was then evaluated using the F1 score and the exact match accuracy of the model over a validation set. These outputs are presented as the results for this section. There were 8 class labels, in which there were some classes that had higher representation. The highest class label corresponded to the 1850-1900 time period, which roughly corresponds to the Romantic period of music. The class represents 25 percent of the samples.

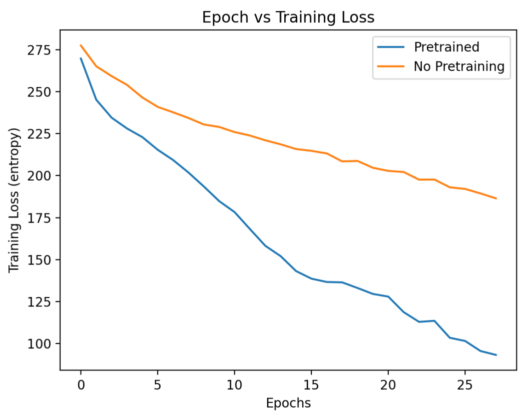

Figure 5: This figure captures the difference in the training loss between using the pretrained model and training model from scratch. It can be observed that the two models start off with relatively the same training entropy. However, the model that uses the weights of the pretrained LSTM model performs converges lower than the model starting training from scratch. This could show that the pretraining task is indeed learning some deep characteristics of music.

4.4 Results and Analysis

The results of the downstream task show that the pretraining done earlier has a large improvement in classification. For the purposes of these experiments, we trained a model using the same sequential input. The difference is that this model doesn't do the generative pretraining task (or the LM task). Instead it trains itself solely on the signal of the time period class. We show that the pretraining task improves upon this model, which has been shown to be effective in this domain. We call this model the baseline model. Figure 5 illustrates that using the weights from the language model for classification can greatly improve the learning of class labels. This means that it is likely that the pretraining task highlighted earlier is indeed effective and is capturing important relationships between pitch, sound, and their relative location to each other. Furthermore, it means that doing this pretraining task can boost the performance of the model, as it is able to understand long term relationships in music. Table 3 illustrates that the model with the pretrained head (LSTM and embedding weights of the LM), does much better than a model that is not pretrained. The table shows out performance in both accuracy and the F1-score of the procedure described in this paper. This would mean that the LM training task has worked and that it allows for us to capture important interactions. We are able to use our pretraining task to improve accuracy and the F1 score.

5 Future Work

In this project, we mainly explored the language modeling for music notes and how does a pretrained language help with the down-stream task. Although we have some interesting observations, there are still many things could be done in the future work. (1) To what extent the music generation task could be done with a language model? We plan to get the architecture of the SOTA language models like GPT-2 (Radford et al., 2019) into play. (2) Human evaluation of the generated music. The perplexity and the per-step accuracy are not enough to evaluate how good a generated piece of music is, some wrong generation of notes may make the music sounds super weird. (3) For the classification, what parts of the training data affect the model decision most? We can utilize techniques like influence function (Koh and Liang, 2017) to understand what time period or which composer affects the prediction most – whether it is consistent with the human judgement.

6 Conclusion

In this project, we explore the use of language model for music generation and use the pretrained language model to aid a downstream task, namely time period classification for classical music. We use the GiantMIDI dataset for both training and testing, the experimental results show that we can get a reasonable generation performance through language modeling of the music notes. Also, by applying the pre-trained language model to the downstream task, we see it help the training in few-shot learning scenario.

References

M. Cuthbert and C. Ariza. 2010. Music21: A toolkit for computer-aided musicology and symbolic music data. In *ISMIR*.

Michaël Defferrard, S. Mohanty, S. Carroll, and M. Salathé. 2018. Learning to recognize musical genre from audio. *ArXiv*, abs/1803.05337.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. Language modeling with deep transformers. In *INTERSPEECH*.

K. Kempfert and S. W. Wong. 2018. Where does haydn end and mozart begin? composer classification of string quartets. *Journal of New Music Research*, 49:457 – 476.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*.

Qiuqiang Kong, Bochen Li, J. Chen, and Yuxuan Wang. 2020. Giantmidi-piano: A large-scale midi dataset for classical piano music. *ArXiv*, abs/2010.07061.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

P. Sadeghian, Casey Wilson, Stephen Goeddel, and A. Olmsted. 2017. Classification of music by composer using fuzzy min-max neural networks. *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 189–192.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

H. Verma and John Thickstun. 2019. Convolutional composer classification. *ArXiv*, abs/1911.11737.