Sequential Modeling of Classical Music Generation

Jiasheng Ni
Center of Data Science
jn2294@nyu.edu

Qirui Zhang
Center of Data Science
qz2519@nyu.edu

Yuan Tian
Center of Data Science
yt2077@nyu.edu

Abstract

In this project, we want to investigate whether sequential models (previously implemented in Kalman Filter or RNN, LSTM) could serve as good modules for existing music generation tasks. In other words, we want to know if the generation results from these models trained causally on the former notes are close to the ground truth notes in the future. Also, we are interested in comparing the generation performance between Kalman Filtering as an unsupervised algorithm with LSTM as a supervised one.

1 Introduction

Music generation and analysis have emerged as pivotal areas of research in computational musicology and artificial intelligence. The inherent complexity of music, with its intricate interplay of rhythm, melody, harmony, and dynamics, presents significant challenges to modeling and synthesis. Recent advancements in supervised learning and large language models (LLMs) have shown promise in automating aspects of music composition and analysis. However, these approaches are often limited by their reliance on large labeled datasets, high computational cost, and inability to adapt seamlessly in real-time scenarios. As such, there remains a compelling need for unsupervised methods capable of addressing these challenges while offering flexibility and computational efficiency.

The Kalman Filter, a mathematical framework originally developed for optimal estimation in dynamic systems, has proven to be an effective tool in analyzing and modeling time-series data. Its application to music processing is particularly intriguing due to its ability to infer latent states from noisy observations, making it a robust choice for modeling temporal structures inherent in music. By leveraging the Kalman Filter's strengths, this study explores an unsupervised approach to music generation that emphasizes adaptability, real-time capabilities, and effective handling of noisy inputs.

1.1 Literature Review

The use of Kalman Filters in music signal processing has been explored in various contexts, with a focus on its utility in modeling and tracking dynamic aspects of music. Satar-Boroujeni et al. [1] introduced a Kalman Filter-based method for tracking partials in music signals. Their work emphasizes the importance of accurate peak detection in spectral representations to capture frequency and amplitude variations over time.

Woskob et al. [2] extended the application of Kalman Filters to the domain of rhythm and beat detection. Treating the problem as a partially observable Markov decision process (POMDP). The Kalman Filter was used to update beliefs about tempo and rhythm in real time, with parameters optimized offline through gradient ascent. This work highlighted the effectiveness of probabilistic filtering in detecting rhythmic structures and adjusting to variations in tempo.

However, few literature really touches on such unsupervised learning model on music generation task, which motivates our works.

1.2 Our Works

This study explores Kalman Filtering and LSTM models for music generation, highlighting their strengths and limitations. Kalman Filtering proves to be a strong candidate, delivering competitive acoustic results similar to LSTM models. While its initial guess influences early note generation and requires more time to converge with random initialization, it excels in adaptability, aligning with music patterns even from poor estimates. Its lightweight design, real-time prediction, and ability to handle tempo changes make it ideal for dynamic settings, though it faces challenges with parameter sensitivity, weaker long-term predictions, and linear assumptions.

LSTM models, on the other hand, excel at capturing long-term dependencies and generating complex harmonies but come with high computational costs and training complexity. Additionally, we observe that standard metrics often fail to align with human perceptions of musical quality, underscoring the need for subjective evaluations or improved metrics. This study highlights the trade-offs between Kalman Filtering's real-time adaptability and LSTM's complexity, providing insights into their application in sequential music generation.

2 Methodology

2.1 Dataset

2.1.1 Terminology

Music theory includes several key concepts that we build upon in our project:

1. A **chord** is a group of notes played together.
2. A **note** is the basic unit of music, representing pitch and duration.
3. A **scale** is a sequence of notes ordered by pitch.
4. **Octave** refers to the interval between one musical pitch and another with double its frequency.
5. **Pitch class**: Number representation of a note within an octave, ranging from 0-11 (e.g. C=>0, C#=>1, D=>2, . . . , B=>11)

2.1.2 Dataset Description

This dataset comprises MIDI files of classical piano compositions by 19 renowned composers, originally curated from the website `http://www.piano-midi.de`. The dataset features a selection of 295 MIDI files, encompassing a variety of pieces and movements from composers such as Bach, Chopin, Mozart, and Schubert among others.

The MIDI files in this dataset represent a wide range of classical music styles and periods, providing a rich source of data for analyzing musical patterns, styles, and structures across different composers and historical contexts. Each file corresponds to a specific musical composition, with some files containing individual movements or sections of larger works. This diversity allows for a comprehensive study of the elements that define classical piano music.

The use of MIDI files is particularly advantageous for musical analysis as they contain precise information about the notes, timings, and dynamics of the performance, making them ideal for detailed computational analysis aimed at understanding and modeling classical music composition and performance practices.

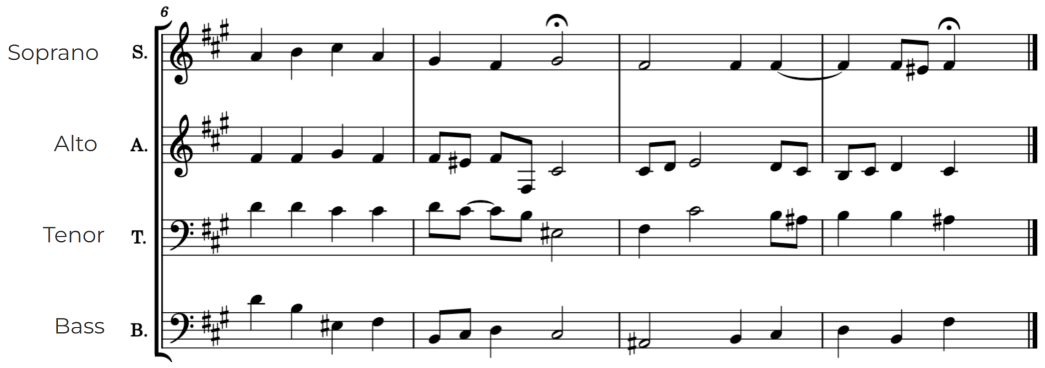


Figure 1: Midi file overview on the computer

2.1.3 Dataset Preprocessing

To preprocess the data, we followed the pipelines below: Chordify the notes belonging to different tracks so that they are grouped onto the same stave (five-line music score);

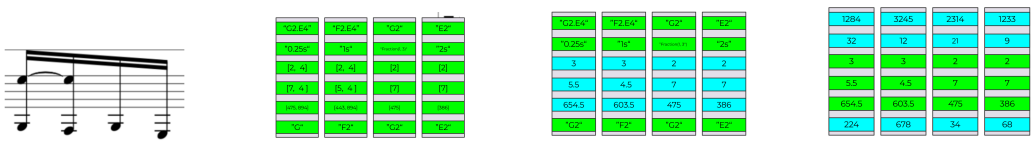
1. Use music-21 library to read midi file into a sequence of music events object, e.g. music21.note.Note, music21.note.Chord.
2. Concatenate notes that are played at the same time from different vocal parts as chords.
3. Extract 6-dim feature out of the music object, including string representation of notes, durations and root notes, number representation of octave, pitch class, and frequency
4. Create the corpus for notes and duration representations so that each string representation of note or duration can be mapped to a unique vocabulary index.

2.1.4 Encode notes and chords

One challenge we encounter while creating number representation for octave, pitch class and frequency is that a chord is composed of simultaneous notes. So, a chord is represented with a list of octave, pitch class or frequency. Thus we need to find a way to invertibly transform a list of number to a single long integer. Currently we use a base-31 transformation so that [34, 23, 334] is transformed by calculating $34 + 23 \times 31 + 334 \times 31^2$=321721. But for frequency, this could produce huge numbers, introducing numerical instability when training the Kalmain Filter. So we perform a feature-wise standardization(which is invertible operation). The final vector representation for each music event is a six-dimensional vector that captures the most frequently used features, including note alias, duration alias, octave, pitch class, frequency, and root note alias.

2.1.5 Pipeline

The pipeline is shown below:



(a) Chordification (b) Feature extraction (c) Chord Flattening (d) Vocab embedding

Figure 2: The pipeline to process the music sequence data

2.2 Modeling

2.2.1 Kalman Filtering

For the model structure, we implemented a standard Kalman Filtering with Expectation Maximization(EM) algorithm. The latent state transition state represent underlying musical patterns that progress over time. The observation state modeled observed data, music event in this case, as a linear transformation of the latent states. Both latent state and observational state contain noise parameters.

Mathematically speaking, we assume that each musical event evolves linearly with the following assumptions:

$$\mathbf{z}_t = \mathbf{A}\mathbf{z}_{t-1} + \mathbf{w}_t$$
$$\mathbf{x}_t = \mathbf{C}\mathbf{z}_t + \mathbf{v}_t$$

where $\mathbf{w}_t$ and $\mathbf{v}_t$ follows multivariable gaussian distributions parametrized by $\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q})$ and $\mathbf{v}_t \sim \mathcal{N}(0, \mathbf{R})$. Each observed music events is preprocessed to be a six-dimensional vector $\mathbf{x}_t \in \mathbb{R}^6, \mathbf{z}_t \in \mathbb{R}^6$ and $\mathbf{x}_{t,i}, \mathbf{z}_{t,i} \in \mathbb{R}, i \in \{1, 2, 3, 4, 5, 6\}, t \in \mathbb{N}$.

For the training and prediction pipeline, the Kalman filter estimates the posterior distribution of the latent state given all observations up to the current time step. The smoother refines the estimates of the latent states by incorporating all observations (both past and future). It uses a backward pass through the time series to compute smoothed mean and smoothed covariance. The EM algorithm is used to optimize the model parameters based on observed data. Finally, using the trained model, the Kalman filter predicts future latent states and corresponding observations. The prediction propagates the latent states forward in time using the learned transition model.

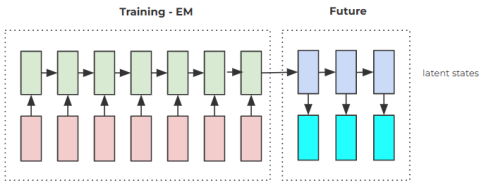


Figure 3: Kalman Filter Model Structure

2.2.2 LSTM

We have developed a music generation model utilizing Long Short-Term Memory (LSTM) networks, implemented through the PyTorch framework. The primary objective of this model is to understand and generate complex musical patterns that capture the subtleties and dynamics of music. The model employs a dual-layer LSTM architecture, further enhanced with an optional attention mechanism, enabling it to effectively handle sequences with intricate melodic and rhythmic structures.

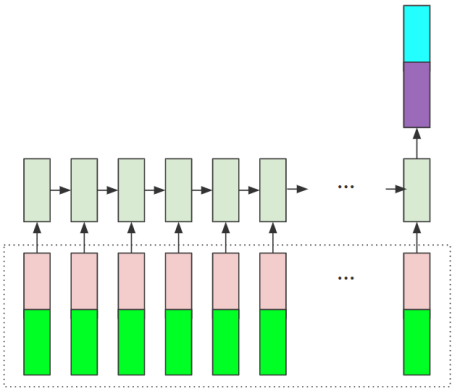
The first LSTM layer focuses on extracting basic temporal patterns from the input sequence, such as short-term note progressions and timing details, while the second LSTM layer delves deeper into these representations to accommodate long-term musical dependencies, such as evolving melodies, harmonic contexts, and rhythmic themes. The collaborative work of these layers allows the model to capture and generate music segments with complex structures.

The added attention mechanism enables the model to concentrate on specific parts of the musical sequence that are more relevant for predicting future notes. This is especially useful in longer sequences where the relevance of earlier notes may diminish over time. The attention mechanism dynamically adjusts the model's focus, enhancing its ability to produce music that maintains thematic consistency over extended periods.

For training, the data is prepared by dividing a comprehensive dataset of musical compositions into an 80/20 training and validation split. This strategic division ensures that the model not only learns but also performs well on music it has not encountered during the training phase. The training process involves optimizing parameters using the Adam optimizer, known for its efficiency in handling sparse gradients and adjusting the learning rate during training. The cross-entropy loss function is used to

```
MusicGenerationNetwork(
  (note_embedding): Embedding(1285, 100)
  (duration_embedding): Embedding(11, 100)
  (lstm1): LSTM(200, 256, batch_first=True)
  (lstm2): LSTM(256, 256, batch_first=True)
  (attention_dense): Linear(in_features=256, out_features=1, bias=True)
  (pitch_out): Linear(in_features=256, out_features=1285, bias=True)
  (duration_out): Linear(in_features=256, out_features=11, bias=True)
)
```

(a) LSTM Structure



(b) LSTM Illustration

Figure 4: The pipeline to process the music sequence data

measure the model's performance by comparing the predicted distribution of notes and durations with the actual distribution.

During training, a patience mechanism is implemented to prevent overfitting by stopping the training process if the validation loss does not improve after a predetermined number of epochs. This is crucial to ensure that the model generalizes well to new data and does not merely memorize the training set. Loss metrics for both the training and validation phases are carefully recorded and monitored to adjust the training strategy as needed.

After successful training, the model's ability to generate new musical sequences is evident in its capacity to produce not only technically accurate music but also pieces that are stylistically varied and creatively inspired. By studying the works of classical music masters and mastering their compositional styles, the model can predict and create new musical movements. Leveraging deep learning techniques and the inherent advantages of LSTM in sequence modeling, this model provides a powerful tool for exploring new musical possibilities and enhancing creative processes in music production.

3 Experiments

3.1 Results

3.1.1 Evaluation Metrics

After some research in music theory, we finalized our choice of five metrics to be the following:

1. **Polyphony Rate**[3]: Defined as the ratio of the number of time steps where multiple pitches are on to the total number of time steps. In our settings, it means the proportion of time during the piece when more than one note is played simultaneously.
2. **Pitch Entropy**: Defined as the Shannon entropy of the normalized note pitch histogram. It measures diversity of notes/chords over all octaves
3. **Pitch Class Entropy**: Defined as the Shannon entropy of the normalized note pitch class histogram. In our settings, it captures the diversity of notes/chords within one octave.
4. **Scale Consistency**[4]: Defined as the largest pitch-in-scale rate over all major and minor scales. It describes whether all notes are within a scale and whether there is any dissonance. Mathematically, it is

$$SC = \max_{\text{root , mode}} \text{pitch\_in\_scale\_rate(root,mode)}$$

where pitch in the scale rate is the proportion of notes that's in the scale determined by root and mode ("major" or "minor").

5. **Groove Consistency**[5]: Defined as the mean hamming distance of the neighboring vector representation of music events. Mathematically, it is

$$GC = 1 - \frac{1}{T-1}\sum_{t=1}^{T-1} d(\mathbf{x}_t, \mathbf{x}_{t+1})$$

where $\mathbf{x}_t \in \mathbb{R}^n$ and $d(\mathbf{x}_t, \mathbf{x}_{t+1}) = \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_1 = \sum_{j=1}^{n} |\mathbf{x}_{t,j} - \mathbf{x}_{t+1,j}|$
Higher values indicate freedom and improvisation and low values for repetition and structures.

3.1.2 Interpretations of Results

Model	Polyphony Rate	Pitch Entropy	Pitch Class Entropy	Scale Consistency	Groove Consistency	Composite Score
Kalman Filter_846	**0.5957**	**2.8165**	**2.2504**	**0.9392**	0.9801	**0.0199**
LSTM_846	0.9901	4.6050	3.2359	0.8488	**0.9992**	0.1136
Original_846	0.7400	2.8748	2.3687	0.9272	0.9996	0.0000
Kalman Filter_847	0.5532	4.7593	**3.2390**	0.7296	0.9840	0.0590
LSTM_847	**0.5490**	**4.7046**	3.2754	**0.8368**	**0.9993**	**0.0347**
Original_847	0.1168	4.3958	3.1405	0.8208	0.9918	0.0000
Kalman Filter_850	**0.6601**	**4.2151**	**3.0080**	0.8094	0.8806	0.0652
LSTM_850	0.6796	4.9807	3.3558	**0.8143**	**0.9993**	**0.0573**
Original_850	0.2292	4.3223	2.9975	0.8477	0.9961	0.0000

Table 1: Results for Music Evaluation Across Models

In Table 1, we observe that the Kalman Filtering model performed particularly well in capturing the Pitch Entropy and Pitch Class Entropy of the original pieces. However, the LSTM model outperformed the Kalman Filtering model in two out of three songs based on the composite metric (weighted absolute difference).

Despite this, we found that the composite metric does not always correlate with the perceived quality of the generated music. For instance, while we internally agreed that for *Bach_847*, the Kalman Filtering model effectively captured the tempo and patterns of the original composition, the LSTM achieved better scores metrically. This highlights an important observation: being closer to the original piece does not necessarily guarantee producing a masterpiece comparable to Bach's works.

3.2 Findings

While testing the impact of different parameters on the Kalman Filtering model, we found that the ratio of σ_w to σ_v yields optimal results, both metrically and acoustically, when it falls between 1.38 and 2.22. Extreme ratios can lead to repetitive notes or chords in the generated piece. However, due to the limited size of our sample, further experiments are required to validate this finding.

When we attempted to overfit the model, we observed that it struggled to converge to the original music piece. We suspect this issue arises from the nature of our string index mapping, which introduces an excessive noise-canceling effect. The simple string index mapping is nonlinear, violating the linearity assumption of Kalman Filtering. One possible solution could be to introduce an additional layer of invertible normalization after the mapping.

4 Conclusion

In this study, we examined the use of Kalman Filtering and LSTM models for music generation, focusing on their respective strengths and limitations in handling the sequential and dynamic nature of music.

Kalman Filtering, as a lightweight model designed for real-time estimation and prediction, demonstrated high effectiveness in scenarios requiring quick adaptation. Its ability to adjust to tempo changes in real time makes it particularly valuable for interactive and improvisational music generation. However, its reliance on linear assumptions, sensitivity to parameter initialization, and weaker long-term prediction capabilities limit its ability to fully capture the complexity of musical compositions.

In contrast, LSTM models excel at learning long-term dependencies and adapting to sequence variations, making them better suited for generating intricate and harmonically rich pieces. Nonetheless, their high computational costs and training complexity pose challenges, particularly in real-time applications.

By comparing these two approaches, this study highlights the trade-offs between Kalman Filtering's real-time adaptability and LSTM's ability to model complex relationships. This analysis offers valuable insights into the selection of appropriate models for music generation tasks, balancing the need for real-time performance against harmonic richness and complexity.

5 Future Works

For future reference, we are considering composing a better composite metric score or even conduct random experiment, building an auto-adapted parameter tuning method for Kalman filtering; In the case for the problem of repetitive notes and error tracking, we are thinking to add another layer of normalization for note index mapping to avoid excessive noise cancelling effect. In short, at least one of the following five aspects could be improved:

1. Better Composite Metric Score Calculation (Random Experiment)
2. Auto-adapted Parameter Tuning Method for Kalman Filtering
3. Add another layer of normalization for note index mapping to avoid excessive noise cancelling effect in Kalman Filtering
4. Implement advanced attention mechanisms in the LSTM model

References

[1] Hamid Satar-Boroujeni and Bahram Shafai. Peak extraction and partial tracking of music signals using kalman filtering. 2005.

[2] George Woskob. Finding rhythm in music using kalman filters optimized with local search. December 2019. `https://github.com/aphera/aa228-final-project`.

[3] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, 2017.

[4] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training, 2016.

[5] Shih-Lun Wu and Yi-Hsuan Yang. The jazz transformer on the front line: Exploring the shortcomings of ai-composed music through quantitative measures, 2020.