Building a Dataset for Music Analysis and Conditional Generation

by

Mengyu Yang

Supervisor: Prof. Sageev Oore

April 2021

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Building a Dataset for Music Analysis and Conditional Generation

Mengyu Yang

Bachelor's of Applied Science Thesis

Supervisor: Sageev Oore

April 2021

Abstract

Current deep learning models for music generation are able to create expressive music but offer little control over its output. Additionally, models do not have an understanding of structure within the music, leading to music that sounds like noodling. A main reason for these shortcomings is the lack of annotated structural information provided in datasets, and these attributes are difficult to extract from the original recordings. We introduce a new dataset that addresses these problems. Consisting of classical and jazz piano played by professional musicians, each classical recording contains additional tracks of the melody, locations of beats and phrases, and the chord progressions while jazz recordings contain the bassline, chord progressions, locations of beats, and multiple variations of improvised solos on a single tune. With this dataset, we propose to build state-of-the-art conditional generative models and musical analysis models.

Table of Contents

List of Figures

List of Tables

Introduction

The advancement of deep learning has led to many interesting subfields, with one area being music generation. When composing music, musicians generally reuse and adapt from portions of music they already know, either consciously or unconsciously [2]. Consequently, computer-generated music is an easy tool to draw inspiration from. Still being a relatively nascent topic, however, means work in this area have mostly focused on simply improving the quality of the generated music. Accordingly, commonly used training sets [3, 4] have not been designed for any additional tasks.

However, a common drawback that current music generation models face is the lack of control over its generated music [5]. Music is a highly creative domain, meaning that if neural nets are to be seriously considered as a tool for music creation, artists will require fine-grained control over the music being created to suit their specific needs. To create these controls, systems [6] have successfully used conditional generation, where the training data includes additional labels of musical attributes for the model to draw relationships from. For example, if during inference we want to specify a specific instrument for the model to include, the training data must contain labels specifying that information. Since existing music datasets do not explicitly provide additional labels, current conditional systems have had to rely on extracting them using metadata or heuristics. While this strategy works for more superficial attributes such as composer, genre, or key signature, important aspects that constitute music's fundamental form is very difficult to extract.

We introduce a dataset of recordings from professional pianists to provide a rich collection of expressive music. Most importantly, the dataset will contain expertly labelled data on three fundamental structural aspects of music: beat, phrase, and melody. This additional information gives us the ability to develop models that can be conditioned on these attributes. It will provide state-of-the-art control, allowing users to finely dictate the form and structure of the music they require from the model.

Related Works

The following section details related works relevant to conditional music generation. Throughout, various musical terms and attributes are introduced to provide context and explanation for when these topics are brought up again in relation to the project.

Music Representations

Score Music is usually represented by the Western notation of the 5-bar staff, called a score. Obviously, another method is required in order for music to be parsed by computers and particularly neural networks. With the traditional score, a lot of information is left out. Although scores depict what notes need to be played in addition to other information such as dynamics, pedalling, and tempo, a large part of a musician's job is to interpret the score. The resulting expressive music can have malleability in tempo, called rubato, as well as the musician's own understanding of dynamics and phrasing, where the latter refers to how sequences of notes are shaped to create music that is more expressive and narrative.

MIDI In order for this information to be relayed to a computer, a commonly used symbolic representation is MIDI, a communication protocol comprised of Note_on and Note_off event messages. Within these messages contain information about the note pitch, represented as a unique integer from the set {0,1,...,127}, and the velocity, indicating how loudly the note is played and also represented by integers from the same set. Also included is timing information for the events. However, Huang et al [7] showed that directly using the MIDI representation does not efficiently preserve polyphony, meaning this representation is likely not useful for the purposes of creating a dataset for generating polyphonic music.

Piano Roll Another method of representation is called the piano roll. As seen in Figure 1, each note is represented by a certain row. Whenever a note is sounded, the respective row records this at the appropriate position along the x-axis, representing time. This method is able to represent polyphonic music since, at each timestep, all notes are individually represented. By discretizing time along the x-axis, the piano roll can be easily converted into a matrix that deep learning models can use. However, this format is not efficient, as it tends to create very large and sparse matrices. Addi-

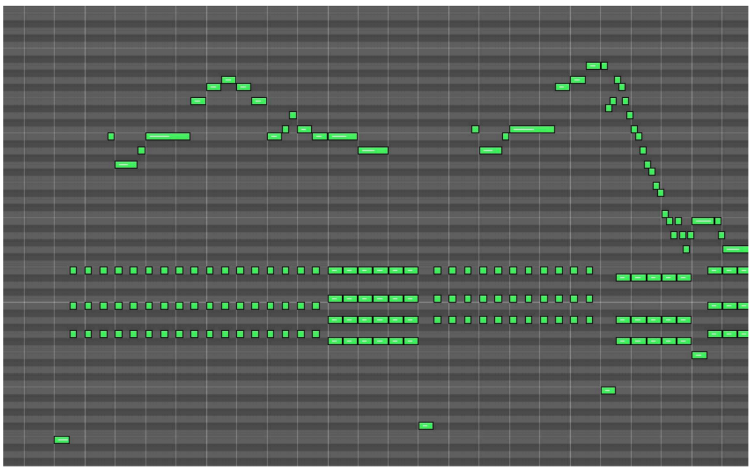

Figure 1: Visual representation of a piano roll, figure credits [1]

tionally, MIDI scores are commonly represented in piano roll format for better visualization.

Performance RNN Oore et al [1] introduced the Performance RNN representation, which models music based on a set of 413 different events. These events fall under 4 categories: Note On, Note Off, Time Shift, and Velocity. The events under each category corresponds to different values that each category can take. For example, the 128 Note On events correspond to the 128 MIDI pitches, where each event signifies the start of a new note. As a result, the Performance RNN representation comes in the form of one-hot vectors, one for each event that occurs in the music. This representation can be further condensed by using an embedding layer, so that each event is represented by a unique integer rather than a 413-length vector.

Although this method specifies note events one by one, it allows for the model to predict an unlimited number of simultaneous notes. Additionally, its ability to express finely quantized time intervals allows for the capture of expressivity in music. This is particularly important, since the generation of expressive and nuanced music is limited in many existing systems [2].

Conditioning

Conditioning refers to the act of providing additional information to the architecture during inference. It is usually given to the model as an additional input and used to provide some level of parameterized control of the output [2]. Examples include providing a bass line so that the output complements it or specifying a music genre for the output to match or an instrument to

include.

To train a conditional model, labels for the conditioned attribute is required. Take for example the case of conditioning on musical genre. During training, a model needs to be given the genre label for each piece of training data for it to learn the relationships between the two. As a result, the crux of building a conditional model is collecting the labels for the attributes that the model is to be conditioned on. While some attributes such as genre and instrument can be easily collected, other attributes are more difficult to label.

As a result, conditional models such as from Meade et al [6] condition on attributes easily extracted from metadata or from using heuristics, such as composer, note velocity, note position, and title keywords. The labelled attributes included in our proposed dataset have not been previously included in a conditional model. This reflects the difficulty in extracting these attributes from existing datasets. However, the proposed attributes contribute significantly to music's fundamental structure, making it an important development towards controllability in music generation.

Datasets

The music domain does not currently have reference datasets [2], unlike that of the image domain. However, there exists a set of commonly used datasets for music generation.

JSB Chorales A longstanding dataset is the J.S. Bach Chorales [3]. However, this dataset is very homogeneous, consisting of only one particular style of music from one composer with the music written originally for voice only. Although Bach chorales tend to sound well when generalized to other instruments [1], the dataset is derived from a score rather than real human performances. As a result, the data is perfectly quantized, making it sound mechanical and without expression. Accordingly, the resulting output of models trained on this dataset will have the same monotonous characteristic.

International Piano-e-Competition Another dataset is from the International Piano-e-Competition [8], which contains MIDI captures from highly skilled pianists. This dataset contains a rich collection of solo classical piano played expressively by experts, which allows for a more realistic output. The data is also coherent, consisting of solo classical piano performances only.

MAESTRO A recent dataset, called MAESTRO [4], builds on the International Piano-e-Competition dataset by also including the finely aligned ($\sim$ 3 ms) audio files corresponding to the MIDI data. As a result, the MAESTRO dataset has the added benefit of facilitating the conversion between symbolic representations such as MIDI and raw audio. This is important because directly generating audio using neural networks is very difficult, but using an intermediate representation provides better results.

However, none of the previous datasets contain explicit labels for conditioning purposes. As a result, systems that currently aim to provide more controllability in its output must derive these labels themselves. Our dataset, in addition to providing a rich and expressive collection from skilled pianists, will include expertly labelled data on musical attributes that target the fundamental aspects of music's form.

Architectures

RNN The recurrent neural network (RNN), as its name suggests, passes previous information from sequential data forward, allowing the model to predict the next element in a sequence based on past information. This past information is encoded in a hidden state, generated at each step in the sequence, and is passed to the model along with the current sequence element.

Since music is inherently sequential, RNNs have been used for music generation. One example is Oore et al's Performance RNN [1], which uses a RNN model trained on the International Piano-e-Competition dataset. The resulting model produces polyphonic music with expressive timing and dynamics.

Despite the convincing music produced by the model, some properties of the architecture can lead to downsides in the context of music generation. First, RNNs process data sequentially, one element after another. As a result, this prevents parallelization during training and reduces the amount of data the model can train on. Another issue related to the RNN's sequential nature is its poor performance with longer sequences of data. Since all past information is encoded into one hidden state, as the RNN progresses deeper into a sequence, information from the beginning can become lost and replaced by more recent information. In other words, RNNs are unable to maintain long-term structure, resulting in the output music sounding like "noodling".

Transformers The more recent Transformer architecture [9] from Vaswani et al addresses the shortcomings of the RNN. The architecture also processes sequential data, but does so as a whole, rather than one element at a time. This particular trait allows for parallelization, leading to more efficient training. It also includes the concept of self-attention and positional encodings, which lets the model highlight different subsets of the sequence depending on what is most relevant without the risk of losing longer-term information.

One particular example of this architecture being used for music generation is Huang et al's Music Transformer [10]. The model modifies the original Transformer architecture by implementing relative positional encodings rather than global, as relative timing is more important in music. This design allowed for the model to generate much longer compositions of around four times the length of Performance RNN. However, neither model allows for control over its output. Incorporating the proposed dataset into the training of these models will allow for both the quality of compositions currently achieved but also the ability to control fundamental aspects of those compositions.

Musical Background

Since this thesis project lies at the intersection of music, some music terms referred to throughout the report are described below.

Monophonic vs. Polyphonic In music theory, monophony refers to music that only consists of a single line of melody with no accompaniment. In other words, there is a maximum of one note being played at any time. On the other hand, polyphony refers to music that simultaneously combines more than one line of music. Examples of polyphony include a melody with an accompanying harmony or two different melodies playing at the same time.

In the context of music generation, monophonic music generation is considered a simpler task, as the more complex the music, the harder it is for a model to learn. It has been shown [1] that the jump from generating monophonic music to polyphonic is a significant task, with many systems making assumptions to lessen the complexities associated with polyphonic music.

Additionally, music representations must be intentionally designed to cater to polyphonic music as well. There are examples, such as Waite's LookbackRNN [11] representation, that can only

represent monophonic music.

Beat In music, the beat refers to the basic rhythmic unit of a measure of music. Different numbers of beats per measure can result in different musical feels. For example, a waltz consists of three beats per measure while most pop songs today uses four beats per measure.

Each beat within a measure can be further classified as either a down beat or weak beat. The down beat describes the first beat within the measure while the remaining beats are weak beats. The down beat is given the most emphasis, which creates a rhythmic pulse throughout the music.

Chord Progression Chord progressions are the foundation upon which most Western music is built upon. It consists of a succession of chords, and these chords provide the defining feature to create the melody and rhythm.

Phrase In music, a phrase is considered to be a unit of music that has a complete musical sense of its own. Often, the analogy between musical and linguistic phrases are made to explain this concept. However, the term is still ambiguous and highly subjective. Different musicians may have different interpretations of where a phrase ends and another begins and both can be correct.

Implementation

This thesis project consists of two parts. The first involves work done on a then current project, Musical Speech. Musical Speech is a demonstration track submission accepted to NeurIPS 2020, and thesis work done for Musical Speech involved building additional components in anticipation for the actual presentation of the system during the conference. The second part of the thesis project, and also considered the main focus, consists of designing and building the aforementioned dataset to be used for conditional music generation and musical analysis.

Musical Speech

Although the main purpose of the thesis project is to design and produce a dataset compatible for conditional music generation, a smaller precursor task arose during the beginning of the thesis term, related to a then-ongoing project. At a high level, the system [12] consists of a tool for music

composition, where it takes as input audio of speech and outputs music based on the speech. The raw speech goes through various layers of processing and neural networks, as outlined below:

1. Extract the fundamental frequencies from the speech audio
2. Map the extracted frequencies to musical notes
3. Apply a sparsifying transformer model that removes notes from the very dense output of the previous step while still maintaining its phonic characteristics
4. Apply a gap-filling transformer model that augments the sparsified sample, making it more musical

While Step 2 can map the frequencies to any instrument, the main demonstration uses piano, given the instrument's popularity among datasets. Accordingly, the output of the system consists of monophonic piano. Given the system's compartmental design, it was desirable to develop additional layers to the system. If the current monophonic output is considered to be a form of melody, it made intuitive sense to develop a Transformer model that outputs a harmonized version of that monophonic sequence. This augmented Musical Speech system is illustrated in Figure 2.

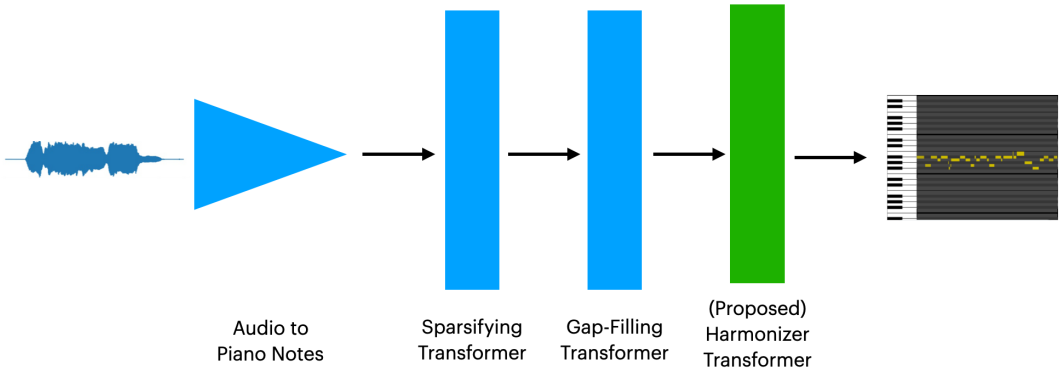

Figure 2: Illustration of the new Musical Speech system. The original system consists of all the blue modules while we propose to add a harmonizing Transformer model to the end. This will take the monophonic output of the original system and harmonize it into a polyphonic one.

The process for completing this task, based on [10]'s experiments with harmonization, was complementary to the main goal of the thesis project. It first involves building a dataset, with the input data being a monophonic melody and the corresponding target being a harmonized version of that melody. The second step involves training a harmonizing Transformer model on that dataset. As a

result, these tasks provided a hands-on opportunity to explore and experiment with different methods of data processing and model training, with the ultimate goal of becoming more efficient and knowledgeable when taking on the main project.

The harmony dataset consisted of solo jazz piano in order to match the instrument of the system. Jazz was chosen because the current output of the system was quite free-form and unstructured, making it more similar in style to jazz than classical. Around 24 hours of solo jazz piano music was collected from YouTube, with artists including Bill Evans, Thelonious Monk, and Art Tatum. The raw audio was then split into 15 second clips, since individual songs were too long for a Transformer model to process. To make the data readable by the Transformer model, the audio clips were transcribed into the MIDI format using Onsets and Frames [13], a state-of-the-art model for piano music transcription.

After this step, we now have the targets of the dataset, consisting of polyphonic piano with melody and harmony. To extract the accompanying melody that serves as the input data, an algorithm called skyline was used. The skyline algorithm is a heuristic for extracting melody from polyphonic music. Simply, it assumes that for all timesteps, the highest frequency note being played is the melody. While this may not be true from a theoretical point of view, this heuristic was suitable for our purposes because our goal is to teach a model to generate harmonies for any given monophonic input, regardless of whether that input is a true melody or not.

Finally, all the MIDI files were then converted into the Performance RNN representation to make it suitable for training purposes. In the end, the dataset consisted of 5163 samples of 15s clips of monophonic jazz piano with the corresponding polyphonic harmonized excerpts. The audio and MIDI formats of the data were kept as well.

The next step was to train a Transformer model on this newly built dataset. The same Transformer architecture as the Gap-Filler model was used for this task. However, the model was only designed for input and output sequences of the same length. In our case, the length of the two sequences differ, since there are more Performance RNN events in a sequence of melody+harmony than just the melody by itself. As a result, padding tokens were used to normalize the difference between the two lengths and these tokens are set so that the model ignores them when encountered.

During initial tests, the model was able to memorize a small sample of data, usually done as a check to show that the model was working. However, despite multiple experiments with training, the model failed to generalize to the entire dataset. The first experiment involved reducing the number of Performance RNN events to only include note on and note off. This reduces the number of different tokens the model must learn and potentially reduce some noise as well. The second experiment involved using a different representation for the input melody altogether, Waite's monophonic Lookback RNN representation [11]. This representation is structured temporally in a grid-like manner, dividing time into small discretized sections. It also uses note on and note off events, so that each discrete unit of time takes on an event value depending on what note, if any, is being played.

Despite both adjustments, the Transformer model still failed to generalize to a larger dataset. Further analysis of these results are presented in the Discussion section.

Conditioning Dataset

Dataset Design This brings us to the second and main portion of the thesis project. Based on current shortcomings of generative and analysis models for music, we introduce the design of a new dataset of expertly played solo piano music accompanied by a rich source of corresponding musical information. Current datasets usually only contain the music itself, either in the form of raw audio or also with accompanying MIDI files. However, many important musical attributes key to producing expressive and controllable music are very difficult to extract from these datasets. Current methods usually involve heuristics or highly hand designed algorithms.

The first step of the design process involves selecting the attributes to be included in this dataset. There are many different musical attributes that models can be conditioned on or taught to analyze but practically, not all can be included. This leads to the need to carefully select a small number of attributes that define the most foundational aspects music is built upon. The dataset also contains both classical and jazz genres. Due to the differences between the two, the musical attributes between the two genres will be different. Tables 1 and 2 shows the attributes included for either genre. These attributes define the main structural aspects of music and by building models conditioned on these attributes, the user will be able to compose a rich variety of music.

Classical Tracks	Description
Original	Recording of the piece as is.
Melody	The main melody of the piece, up to the musician's interpretation.
Countermelody	The secondary melody. Silence if not applicable.
Phrases	One note to mark the start of a phrase and a different note to mark the end of a phrase.
Beats	One note to mark the down beat and another note to mark the weak beats.
Chord Progression	The chord of the progression is held for its duration.

Table 1: Tracks included in the classical recordings and their descriptions.

Jazz Tracks	Description
Bassline	A walking bassline, can be looped for multiple choruses.
Beats	One note to mark the down beat and another to mark the weak beats.
Chord Progression	A combination of chords held for the progression's duration and more musical comping. Can be looped for multiple choruses.
RH Melody+Solo	Only the right hand plays first the head of the jazz tune and then an improvised solo.
Solo w/ Bassline	Improvised solo with both hands playing. The solo is played while listening to the bassline track, leaving the left hand to play more chords and solos.
Solo w/o Bassline	Improvised solo with both hands playing. The solo is played without listening to the bassline track, allowing the left hand to play more bass-like patterns.

Table 2: Tracks included in the jazz recordings and their descriptions.

The dataset consists of MIDI files, since it is a symbolic representation that can be converted into vector representations. This means that each recording consists of a multi-track MIDI file, where the first track is the original recording and the other tracks consists of each attribute that have been approximately temporally aligned. This alignment is naturally achieved during the creation of each track, as the musician will listen to the original track and record each attribute track over it. A visual example of a MIDI file is shown in Figure 3.

Data Collection Now that the structure of the dataset is established, the next step involves actually collecting the data. Due to the nature of remote work, the required equipment must be commonly possessed by pianists, meaning no specialized equipment or software. As a result, our dataset only requires a keyboard with MIDI output, a computer to connect the keyboard to, and any digital audio workstation (DAW) that can export MIDI files.

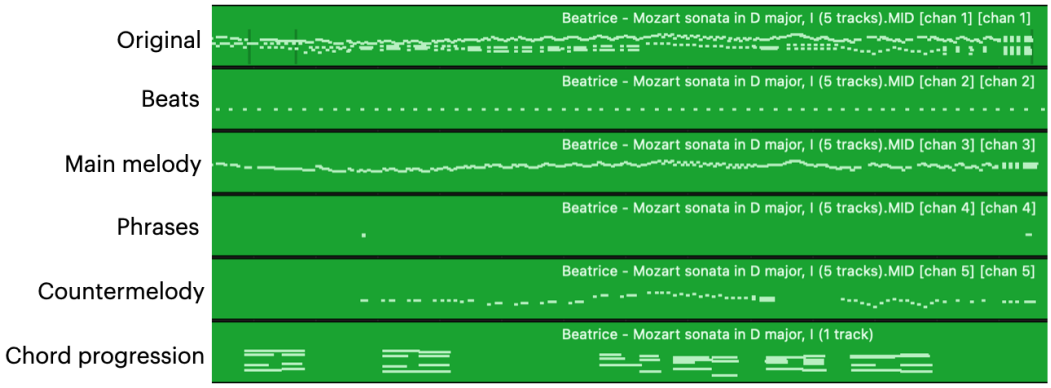

Figure 3: Example of a section of a MIDI recording from the dataset, presented in a piano roll style. Note the multi-track format and the alignments of each track due to the play-back-and-record methodology.

To begin, the musicians first complete the original recording by playing the piece as is. After finishing this first track, they will then play back the piece and record the remaining tracks while listening to the original track. Sometimes, the piece may be too fast and the musicians may find it difficult to record the other tracks during play back. This problem is especially prominent when they are sight reading pieces. To overcome this, musicians were advised to leverage the MIDI format and play back the original recording at a slower tempo, which can be easily done from the DAW interface. Once the attribute tracks are recorded at a slower tempo, the tempo for all the tracks can then be adjusted back to the original value.

Data collection during remote work provided additional difficulties in getting everything running. Communicating musical ideas is a highly practical process. It is much easier to simply demonstrate rather than try to explain. However, this becomes difficult when people are situated across the continent. This led to the first recordings from the musicians being an iterative process, involving multiple revisions until both parties were on the same page. However, as we collected more data, we were able to show examples to new musicians, which made onboarding much easier.

Dataset Helper Functions

Given the raw data, some additional processing is required for the classical recordings. Namely, the phrase and beat markers need to be properly aligned, as the musicians can make mistakes and play the markers slightly off. Additionally, given either the melody or countermelody track, we need to know which notes in the original track correspond to melody. Only with this information

can we map a recording to its melody or vice versa, which is needed to train models for melody extraction or harmonization.

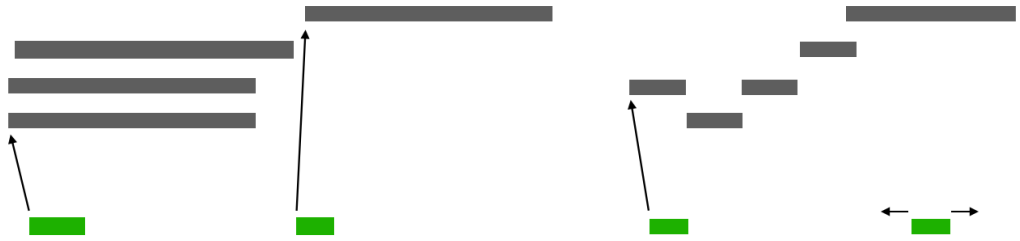

Figure 4: Illustration of the marker alignment helper function. Notes are presented in a piano roll fashion where grey notes belong to the original track and green notes are the beat or phrase markers from the other track.

Marker Alignment Figure 4 illustrates the algorithm for aligning the markers. For a particular marker m, let the start time of m be t. Given a predefined threshold j, the function looks within the range $[t - j, t + j]$. If any note from the original track starts within this range, then the start time of m is adjusted to the same value. If no note from the original track start in this range, then the marker is left as is.

Figure 5: Illustration of the melody alignment helper function. The black dotted lines show the alignment between notes in the melody track and the corresponding notes in the original track.

Melody Alignment Figure 5 provides an illustration of the melody alignment function. At a high level, it identifies the notes in the original track that align with the notes found in the melody tracks. The function is based on the dynamic time warping (DTW) algorithm but with a custom cost matrix, since usual methods of calculating distance cost such as Euclidean distance cannot be used on MIDI notes. As a result, we define our own cost function for MIDI notes. Let i, j be two MIDI notes and $\mathcal{L}(i, j)$ be the function that computes the distance cost between the two notes. Then,

$$\mathcal{L}(i,j) = \mathcal{L}_{value} + \mathcal{L}_{start} + \mathcal{L}_{duration}$$

where

$$\mathcal{L}_{value} = |i.value - j.value|$$

$$\mathcal{L}_{start} = |i.start - j.start|$$

$$\mathcal{L}_{duration} = |(i.end - i.start) - (j.end - j.start)|.$$

In words, the cost function is the sum of three components: the absolute difference in MIDI note value, the absolute difference in start times, and the absolute difference in note duration. Using this function, we construct the $m \times n$ cost matrix C, where m is the number of MIDI notes in the original track, n the number of MIDI notes in the melody track, and $C_{i,j} = \mathcal{L}(i,j)$, with i and j coming from different tracks.

Once C is computed, the standard DTW algorithm is run. This results in two lists of length m, O and M. O is related to the original track and is made up of $[1, 2, 3, ..., m]$, corresponding to each note index within the original track. The elements of M refer to the note indices of the melody track and has repeated values. For each element in O, the corresponding element in M is the closest matching note found in the melody track. Take for example $M = [1, 1, 1, 2, 2, 3, ..., n]$. Then this means that the first 3 notes in the original track all match closest with the first note in the melody track.

But we require a one-to-one mapping, which currently this algorithm does not supply. In our example, we have 3 notes from the original track which all match to one note in the melody track. We need to find which pair out of the 3 match the closest. Doing so simply involves looking up the distance cost value of each pair in C. In other words, the closest-matching pair is determined by $\min\{C_{1,1}, C_{2,1}, C_{3,1}\}$. This is then repeated for every note in the melody track.

Results & Discussion

Harmonizing Transformer

Previously, we discussed how despite multiple attempts, the harmonizing Transformer model was unable to generalize to a larger dataset. Here, we discuss potentials reasons for why that may be. The first likely reason is due to the use of the skyline algorithm for melody extraction. Coupled with the nature of solo jazz piano along with the noise introduced when transcribing raw audio to MIDI using Onsets and Frames, the extracted melody at many times did not sound ”correct”. Instead, there were lots of random notes interspersed throughout, likely from when only the left hand is playing short chords and also when noise from the raw audio gets transcribed into notes.

Another reason is likely the poor mapping that exists between the input melody and the target sequence. Since the skyline algorithm only cares about the highest frequency note, it may only identify a portion of a note as belonging to the melody. Combined with the randomly occurring notes, it becomes difficult for the model to learn a clean mapping between the input and output sequences.

Both of these reasons introduced difficulties in getting the Transformer model to generalize to a larger amount of data. However, this task can be revisited with the new conditioning dataset, which has no noise since it is directly recorded in MIDI and the melodies are identified by expert humans rather than simple heuristics.

Conditioning Dataset

Having overcome the difficulties in starting data collection, building the dataset mostly consists of supervising over the incoming data to ensure quality and proper format. Based on the data currently collected, recording mostly happens in real-time or faster. For classical, the original track can be sped up when recording sparse attribute tracks such as beats or phrases. For jazz, the bassline, beat, and chord progression tracks can all be looped for multiple choruses.

The rate of data collection is highly dependent on the number of musicians hired as well as their availability. As we go into the summer term, we foresee a faster rate of data collection as more musicians become available for a larger amount of time. Additionally, as we collect more data,

small models can be built to overfit on the small dataset in order to test different architectures and representations of data.

Helper Functions

Evaluation of Marker Alignment When testing the marker alignment helper function on collected data, we found that in all test cases, the alignment achieved purely by the musicians were accurate enough that the adjustments made afterwards by the helper function were negligible. On average, notes were adjusted on the order of tenths to hundredths of a second, which is practically indiscernible to the human ear. But while this helper function does not seem to be as useful for the current collection of data, its use may be required once musicians begin sight reading pieces and recording markers become a more reactionary process.

Evaluation of Melody Alignment The melody alignment function was tested qualitatively on some of the collected data as well. We found that the function performs very well when the melody is monophonic, achieving 100% accuracy in identifying all melody notes in the original track. The function is even able to correctly isolate all trilled notes within the melody, meaning the function is able to isolate very short and rapidly-changing notes.

When encountering chords in the melody track, there were cases when the extracted melody in the original track loses a note from the chord. This is most likely caused by the restriction with DTW that mapped indices must be monotonically increasing. This means if note i from the original track maps to note j in the melody track, any note after i cannot map to a note before j and vice versa. Since chords are still represented as a single sequence in MIDI and its notes are ordered by the very small differences in start times, the order of notes between two recordings of the same chord may be different, and that difference can prevent the mapping of notes in that chord. As a result, additional work needs to be done in pre-processing both recordings to ensure the ordering of notes is the same.

Finally, we also tested the robustness of the melody alignment function by testing melody extraction on a completely different recording of the same piece. Since the interpretation is different, the temporal alignment between the melody and original track is much poorer than before. In our

test case where we ran the helper function on the new recording without any pre-processing, the function was able to extract most of the melody notes, with around 75% accuracy. This drop in accuracy shows the need for additional pre-processing to first adjust the tempo of the recording to match the melody track as closely as possible. Also, because the alignment of the tracks is not as precise, the weightings of the components of the cost function also needs to be adjusted to reflect this. More weight needs to be assigned to differences in note value and note duration and less weight to note start time.

Conclusion

After highlighting the shortcomings of current music datasets and music generation models, we propose a new dataset for conditional music generation and music analysis. In addition to containing expertly played classical and jazz recordings, each recording also contains additional tracks that annotate important structural information of the music. These attributes, with the help of several helper functions, are finely aligned, meaning it is easy to map the additional information to the original recording. Once enough data has been collected, future work can look at building state-of-the-art models using this dataset.

References

[1] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This Time with Feeling: Learning Expressive Musical Performance," *arXiv:1808.03715 [cs, eess]*, Aug. 2018. arXiv: 1808.03715.

[2] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, "Deep Learning Techniques for Music Generation – A Survey," *arXiv:1709.01620 [cs]*, Aug. 2019. arXiv: 1709.01620.

[3] J. Bach, *389 Chorales: for SATB Voices; Choral Score*. Kalmus Classic Edition, Alfred Publishing Company, 1985.

[4] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset," *arXiv:1810.12247 [cs, eess, stat]*, Jan. 2019. arXiv: 1810.12247.

[5] J.-P. Briot and F. Pachet, "Music Generation by Deep Learning - Challenges and Directions," *Neural Computing and Applications*, vol. 32, Feb. 2020. arXiv: 1712.04371.

[6] N. Meade, N. Barreyre, S. C. Lowe, and S. Oore, "Exploring Conditioning for Generative Music Systems with Human-Interpretable Controls," *arXiv:1907.04352 [cs, eess]*, Aug. 2019. arXiv: 1907.04352.

[7] A. Huang and R. Wu, "Deep Learning for Music," June 2016.

[8] "International Piano-e-Competition."

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv:1706.03762 [cs]*, Dec. 2017. arXiv: 1706.03762.

[10] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer," *arXiv:1809.04281 [cs, eess, stat]*, Dec. 2018. arXiv: 1809.04281.

[11] E. Waite, "Generating Long-Term Structure in Songs and Stories."

[12] J. d'Eon, S. H. Dumpala, C. Sastry, D. Oore, M. Yang, and S. Oore, "Musical speech: A transformer-based composition tool," 2020.

[13] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and Frames: Dual-Objective Piano Transcription," *arXiv:1710.11153 [cs, eess, stat]*, June 2018. arXiv: 1710.11153.