Music Synthesis via Chord Generation

BENJAMIN GEYER, George Mason University

ACM Reference Format:
Benjamin Geyer. 2021. Music Synthesis via Chord Generation. 1, 1 (March 2021), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The field of music generation and synthesis has been established for many years but recent efforts through the avenues of machine learning and algorithmic composition have yielded impressive results. Computers have aided musicians from modifying certain sounds via synthesizers, recording and syncing audio from various sources, pitch correction, and even generating musical scores to be performed later by humans. However, this review focuses on more modern approaches using machine learning methods to create music such as through the generation of chords to accompany a melody. This specific subtask is well constrained and even has direct applications from helping musicians compose entire songs from a simple melody to helping educate aspiring musicians on proper music theory practices via the identification of alternate chords for existing songs.

1.1 Brief History

Similar to general machine learning, music generation researchers historically used a variety of Generative Grammars, Markov Chains, and combinatorics before the advent of deep neural networks much later [2–4]. These early approaches were limited in flexibility, variability, long-term coherence, and the degree of control over what was generated.

1.2 Intuition

One of the main issues with generation good music is having the ability to properly evaluate what is good and what is not good music. An approach to solve this is to train models on large amounts of human created data with the intuition being that the human-generated songs have good music theory encoded in them. By having the models train and generate chord progressions which are a specific aspect of music theory, hopefully the models will directly learn some aspect of music theory without humans attempting to encode a biased and limited representation of music theory.

2 PRELIMINARIES

2.1 Musical Preliminaries

- `Bar`: A unit of time containing a certain number of beats in a musical piece.
- `Melody`: A sequence of individual notes that makes up the main distinctive satisfying sound of a song. Many musicians start songs with melodies that are interesting and unique and compose songs by layering chords and other instruments along with the melody.

Author's address: Benjamin Geyer, bgeyer3@gmu.edu, George Mason University, Fairfax, Virginia.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
XXXX-XXXX/2021/3-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- `Chord`: 3 or more notes played at the same time during a song. This typically occurs while a melody is played over the top. Chords also make up the harmony of the song.
- `Chord Progression`: The change in chords over time that determines how a song unfolds. These changes in chord typically are associated with changes in emotion or feeling during a song.
- `Musical Instrument Digital Interface (MIDI)`: A common file format for storing songs using the note information of the song rather than the actual audio of the notes being played. These files are small in size and easily interpreted by a variety of software programs allowing musicians and researchers to collaborate and communicate on music projects without dealing with raw audio. Many of the datasets used in modern music generation deal with training on and outputting MIDI files.

2.2 Architectural Preliminaries

An embedding is a low-dimensional representation of a higher-dimensional vector. Embeddings are common across a variety of machine learning tasks which benefit from mapping complex detailed data into a semantically dense low-dimensional space where similar items are near each other. This can be applied in music generation for embedding the components of the music itself from notes, chords, and instruments to properties of those components such as positional embeddings for notes in a bar. This helps give the models detailed information about each component of a song in a dense and interpretable format which can be learned.

Markov Chains are stochastic models which use probabilities of sequential events based on the previous states. These Markov chains have been used in music generation dating back to 1981 when David Cope began experimenting with Markov Chains, generative musical grammars, and combinatorics [4]. However these techniques face several challenges including the limitation of the model to generate musical subsequences that occurred in the training data resulting in very unoriginal sounding music.

Recurrent Neural Networks (RNNs) seek to solve these problems by modeling a sequence of notes where the output of each node in the network is passed to the next node in the sequence. This closely models time-related data such as music by having previous notes affect later notes. In 1989, Peter Todd first used Recurrent Neural Networks, named sequential networks at the time, on generating music [12]. However, regular RNNs are faced with the issue of the dependence of notes fading very quickly resulting in music that has low global coherence. This is known more broadly in machine learning as the vanishing gradient problem and affects many different areas in sequence learning and composition.

In 2002, Douglas Eck applied the novel Long-Short-Term-Memory (LSTM) Network architecture to generate music with a high level of global coherence resulting in pleasing sounding music [6]. LSTMs are an extension of RNNs which have multiple gates to control the amount of data passed along a sequence to allow for more dynamic control of contextual memory. LSTMs use linear units called Constant Error Carousels (CECs) which are connected to each other along the sequence and pass along relevant data. However, these CECs are modified by multiplicative input, output, and forget gates which work together to maximize the flow of relevant contextual note data along the sequence while minimizing the impact of non-important data diluting the important data.

Bidirectional LSTMs (BLSTMs) are a simple extension of LSTMs that allow long-term dependence in both directions along a sequence. BLSTMs simply have two LSTMs where one is reversed and trains on the sequence in reverse. The two LSTMs are then combined for each timestep in the sequence to give a combined output with context in both directions. This has the added benefit for music generation of giving context to a certain note of a song based on the notes that come before and after it which can improve the global coherence of a generated song.

Convolutional Neural Networks (CNNs) differ from RNNs and are commonly used for computer vision tasks but have recently been applied to a variety of machine learning problems. CNNs work by sliding a matrix, known as a filter, kernel, or window, over the input data and for each matrix position, computing the dot product of the matrix and the contained data in the matrix window. This value is stored in the output convolved matrix where each entry in the matrix corresponds to an application of the kernel matrix to a subset of the image that the kernel matrix spanned. The size of this matrix is controlled by the size of the kernel, the stride of the kernel (how far it moves between each convolution), the padding of the original image/matrix to allow the kernel to reach the edges properly, as well as the number of different kernels applied to the image/matrix. If multiple kernels are applied, then the convolved matrix has a dimension for each kernel applied. The downsides to CNNs for music however, are that CNNs can apply in the time dimension but not the pitch dimension where changes in pitch are not linear and hence a linear dot product convolution would not be proper to use. However, CNNs are faster to train and much more parallelizable than RNNs.

Usually, after convolution, either mean or max pooling is applied to the convolved matrix output. Max pooling is simply sliding a window of size nxn across the convolved matrix and taking the max value for each position of the max pooling window. Mean pooling is very similar with taking the average of the pooling window rather than the maximum value. The benefit to adding pooling after convolution is to reduce the dimensionality of the intermediate representation while preserving meaningful information. Max pooling allows the CNN to act as a feature detector that signals as soon as that small feature has been detected. Mean pooling can help give an accurate scoring of all of the pixels/elements inside of the kernel matrix window according to the values of the kernel matrix.

Attention is a popular modern mechanism to allow a model to emphasize certain elements of the sequence for each timestep based on certain learned weights. This is inspired by the human ability to focus attention on certain objects for tasks such as object tracking and recognition. Attention mechanisms have been demonstrated with great success on a great deal of recent machine learning tasks. One recent technique using attention is called a Transformer which uses exclusively attention connections in place of recurrence or convolutions in an encoder-decoder architecture [13]. Researchers have shown that Transformers could be applied to music but with a few challenges including having to modify the positional encoding algorithm to be based on the relative distance between two elements in the sequence [8].

3 AREA TAXONOMY

3.1 Type

One categorization in the type of music being generated in Music Generation Systems is monophonic, polyphonic, and homophonic music. Monophonic requires only single notes being played with no accompanying harmony, polyphonic requires at least two independent melodies occurring simultaneously, and homophonic requires a melody with an accompaniment in the form of a harmony. An example of a monophonic instrument is a flute, whereas pianos and guitars are classic examples of polyphonic instruments. While monophonic music is both simpler for a model to learn, the vast majority of training data is polyphonic or homophonic, which increases model complexity.

3.2 Objective

The objectives of these generative systems fall into two main categories: Next Note Prediction and Musical Accompaniment. Next Note Prediction is where a model is trained to continue a song given a small sample to start with. Musical Accompaniment, however, is based on the idea of adding

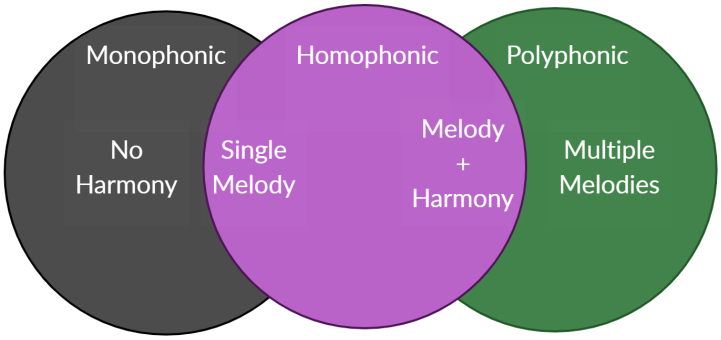

Fig. 1. Venn Diagram showing the similarities and differences between Monophonic, Polyphonic, and Homophonic music.

layers to a given song. Examples of this include adding more instruments or vocals (melodies) to songs, adding melodies to songs with just a chord progression, and adding chord progressions to songs with just a melody.

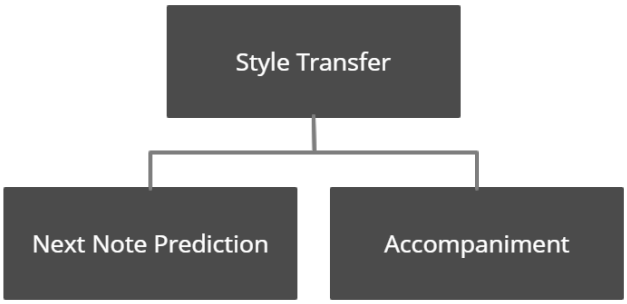

Fig. 2. Hierarchical Diagram showing the relationship between the Style Transfer, Next Note Prediction, and Accompaniment objectives.

However, there are researchers working on models that can do both Next Note Prediction as well as Musical Accompaniment for the greater goal of Style Transfer of artists, genres, and instruments. This task can be done using Next Note Prediction with a sample of a known song, but being told to generate the rest in the style of another artist or genre or with other instruments. Musical Accompaniment can also be done in Style Transfer by adding additional instruments or vocals to a song or generating chords for a song in a specific genre or artist style.

4 TAXONOMY-BASED SURVEY OF THE AREA

4.1 Next Note Prediction

Next Note Prediction is the simplest of the music generation problems to understand and test. It simply means training on small subsets of each song and then testing the models performance on generating each next note properly compared to the actual song. These generated songs can be evaluated in several ways:

- The Top-N note accuracy - for each timestep, the model is correct if the expected note is in the top N notes ordered by probability outputted from the model.

- Cross-entropy - calculate the negative log likelihood of the probability of the expected note being on at the correct certain timestep based on the model's output.
- Calculate the cosine similarity between the still-embedded output from the model and the model-embedded expected song.

Many of the early music generation systems used monophonic music such as David Cope with Markov Chains in 1981and P. Todd with early RNNs in 1989 [4, 12]. This early on, researchers were simply attempting to get the network to learn single monophonic melodies. The problem with polyphonic music early on was the representation of pitch and begin-time of notes being played simultaneously was double the output units which was deemed an unreasonable solution due to computation costs. Douglas Eck also worked on Next Note Prediction early on with a short recording of Blues as a prompt for never ending generated Blues music which he informally claimed was "pleasant" sounding although no statistical evaluation of the quality of musical output was calculated [6]. This is a common problem for music generation as music is not objective and can be very hard to measure objectively.

4.2 Musical Accompaniment

Musical Accompaniment is focused much more on using the computer to augment and improve an already existing song by adding layers. This is done by adding instruments, vocals, melodies, or chords to a song. The model is typically trained by being fed a song with one "layer" stripped away and attempting to generate the missing layer. This allows the model to learn much more complex musical concepts incrementally leading to better results.

4.2.1 Instruments / Vocals. The first aspect of musical accompaniment is adding more instruments or melodies to a song. In general, vocals function as melodies in music which means that adding vocals is the same as adding melodies. This means that the task of adding any new monophonic instrument or vocal can be done with the same model. However, as mentioned before, polyphonic music is much more complicated and as such, adding instruments such as guitar and piano with many notes being played simultaneously in the melody can be tricky for a model to do well.

4.2.2 Chord Progressions. The other category of musical accompaniment is generating chord progressions for a given melody. One interesting aspect of melody to chord progression transitions is that it is a fine to coarse representation transition. This means that melodies are information dense in terms of time whereas chords contain information but at a much coarser level. This means that the melody puts more constraints on the chords being generated. In the opposite case, there are many possible "good" melodies that can accompany a given chord progression. For example, the 12-Bar Blues is a very ubiquitous chord progression for the Blues genre that almost all of the melodies in the genre are derived from. This makes the problem of generating a chord progression for a Blues melody rudimentary for this genre, whereas the opposite is nearly impossible.

The reason why chord generation is important is that many people can come up with a catchy melody or lyric but struggle with the music theory knowledge needed to create a cohesive full-length song. Generating a chord progression to accompany a given melody in either instrumental or vocal form can go a long way to making the piece into a real song. This is the exact motivation behind Microsoft's MySong which generates chord progressions for vocal melodies [11]. Their program uses a combination of a Hidden Markov Model (HMM) and a non-temporal chord probability model that predicts chords based on the notes that appear in specific bars. This approach allows the user to tune the output to depend more on the HMM or the chord-based probability model which is quite a unique feature. Once again, the researchers don't have a more statistical method of measuring

the quality of the generated solutions than having humans compare and grade them compared to chords manually generated by other musicians.

A more modern approach to chord progression generation by Yang. et al. is by using two LSTMs to separately learn a chord-to-note relationship embedding, and a chord-to-chord progression embedding [14]. They compare their model to a traditional HMM model but use a rudimentary evaluation metric of chord repetition rate with the logic that if a chord repeats itself a lot, it is bad. Although their metric seems slightly flawed, their model architecture seems to show some promise. Another dual-LSTM-based paper by Brunner et al. generates chord progressions for polyphonic music and their model even learned to extract the circle of fifths which is a common music theory tool for organizing the 12 chromatic pitches [1]. Interestingly, their model estimates the chords of the song during preprocessing by computing a histogram of the 12 notes over a bar and the three most played notes correspond to the chord. This approximation can work because if a chord is held down, then the notes being held down will count for each beat in the bar. However, there are several issues with this approach: chords can be more than 3 notes, chords can change in between bars, and other notes can be played at the maximum tempo of the bar but with just short duration also counting as equal to the chord notes. Once the detected chord is passed to the network, it is encoded via a learned embedding layer along with the melody of the song. Then the embedded chord of the next time step is appended to the current vector to give it some future context. Additionally, the researchers introduce a 1 to 8 positional embedding for every note on the bar to allow the model to know where along the bar the current timestep is. These complex model architecture decisions along with a solid cross-entropy evaluation function help the researchers achieve fantastic results with the chord embeddings even approximating the circle of fifths after PCA reduction to 2 dimensions. Similar music theory approximation results were also achieved with a Variational-Auto-Encoder with structured attention from researchers Zhao et al. [16].

Researchers Lim et al. [9] compared BLSTMs directly against HMMs, and Deep Neural Network-HMMs (DNN-HMMs) using 25 human scorers as well as an accuracy measure of chord prediction on a test set and found that BLSTMs outperform both with DNN-HMMs slightly outperforming traditional HMMs.

Other researchers have used Digital Audio Workstations to generate simple chord progressions for a given melody and then fed those to a Genetic Algorithm to breed the best chord progressions for a certain melody according to a custom fitness function [15]. This custom fitness function depends on the variation in pitch in the first and second verse of the melody. If the variation is high, "the score takes into account the number of 'base' notes and the denseness of the chord notes" [15]. Alternatively, if the variation is low, the fitness is based on the variation and repetition rates of the chord progression. If it is too low or high as determined by the researchers with hyperparameters, then the fitness is low. This approach can be useful for very specific use cases but strongly biasing fitness functions and hyperparameters can prevent the model from transferring well to other datasets or musical preferences.

4.2.3 Style Transfer. Style Transfer tackles the problem of generating music with unique new combinations of artists, genre, instruments, and lyrics. Because of the complex nature of the problem, most approaches to Style Transfer use BLSTMs or Transformers. OpenAI has two publicly released models MuseNet and JukeBox that both tackle the Style Transfer problem in music generation [5, 10]. JukeBox for example can generate new music given a seed of the first 6 notes of Chopin's Nocturne but using piano, drums, and guitar all in the style of Bon Jovi. The researchers generated a visualization of genre and artist embeddings which demonstrated their model's ability to effectively organize and represent the style differences across different genres and even individual artists.

5 OPEN PROBLEMS

Despite all of this progress, there are still many open problems in the field of Music Synthesis and Generation including global coherence, originality vs imitation of training examples, control over the intermediate algorithm steps, transfer learning, and effective qualitative evaluation of generated music. With the advent of LSTMs and Transformers, global coherence has definitely improved but a quick listen of a few sample songs from OpenAI doesn't indicate that the problem is solved yet at all, even with Transformers. Another problem is the issue of models outputting sequences of notes that are simply directly from sequences in the training set rather than creating new audio every single time. In general, it is hard to guarantee novelty at all with neural network systems. The biggest open problem at the moment is the effective evaluation of generated music samples; even researchers with very promising advanced architectures are still relying on a combination of rudimentary metrics and small amounts of humans grading tiny subsets of generated samples to measure their performance.

6 PROJECT DESCRIPTION

The focus of this project is on Music Synthesis via Chord Generation given song melodies using Deep Learning. Both LSTM and Transformer-based models have demonstrated fantastic results in recent years and both would make good candidates for a model architecture to experiment with. Given computation and time constraints, BLSTMs are likely the better choice. The main factor that is left to be decided is the dataset to test on. Many of the papers use different and slightly customized datasets but MAESTRO V3 from MusicTransformer [7] seems to have many positive traits compared to others used in other studies. Learned chord and positional embeddings along with an embedded melody will be passed into either a BLSTM model or a Transformer architecture and will eventually output a chord progression to accompany the original melody.

REFERENCES

[1] Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Jonas Wiesendanger. 2017. JamBot: Music Theory Aware Chord Based Generation of Polyphonic Music with LSTMs. *CoRR* abs/1711.07682 (2017).

[2] Ruofeng Chen, Weibin Shen, Ajay Srinivasamurthy, and Parag Chordia. 2012. CHORD RECOGNITION USING DURATION-EXPLICIT HIDDEN MARKOV MODELS.

[3] N. Chomsky and D.W. Lightfoot. 2009. *Syntactic Structures*. De Gruyter. https://books.google.com/books?id=mrz3TsgLPzQC

[4] David Cope. [n. d.]. Experiments in Music Intelligence. *In Proceedings of the International Computer Music Conference, San Francisco: Computer Music Association* ([n. d.]).

[5] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. 2020. Jukebox: A Generative Model for Music. arXiv:eess.AS/2005.00341

[6] Douglas Eck and Jürgen Schmidhuber. 2002. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In *NEURAL NETWORKS FOR SIGNAL PROCESSING XII, PROCEEDINGS OF THE 2002 IEEE WORKSHOP*. IEEE, 747–756.

[7] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse H. Engel, and Douglas Eck. 2019. Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=r1lYRjC9F7

[8] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2019. Music Transformer: Generating Music with Long-Term Structure. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=rJe4ShAcF7

[9] Hyungui Lim, Seungyeon Rhyu, and Kyogu Lee. 2017. Chord Generation from Symbolic Melody Using BLSTM Networks. arXiv:cs.SD/1712.01011

[10] Christine Payne. 2019. MuseNet. openai.com/blog/musenet

[11] Ian Simon, Dan Morris, and Sumit Basu. 2008. MySong: Automatic Accompaniment Generation for Vocal Melodies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. Association for Computing

Machinery, New York, NY, USA, 725–734. https://doi.org/10.1145/1357054.1357169

[12] Peter M. Todd. 1989. A Connectionist Approach to Algorithmic Composition. *Computer Music Journal* 13, 4 (1989), 27–43. http://www.jstor.org/stable/3679551

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[14] Wei Yang, P. Sun, and Y. Zhang. 2019. CLSTMS: A Combination of Two LSTM Models to Generate Chords Accompaniment for Symbolic Melody. *2019 International Conference on High Performance Big Data and Intelligent Systems (HPBDIS)* (2019), 176–180.

[15] S. D. You and P. Liu. 2016. Automatic chord generation system using basic music theory and genetic algorithm. In *2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*. 1–2. https://doi.org/10.1109/ICCE-TW.2016.7520919

[16] Yizhou Zhao, Liang Qiu, Wensi Ai, Feng Shi, and Song-Chun Zhu. 2020. Vertical-Horizontal Structured Attention for Generating Music with Chords. arXiv:cs.SD/2011.09078