TyNet: Generating Jazz Harmony using Neural Networks

Jens Honack JHONACK@SAS.UPENN.EDU
Damian Krupa DKRUPA@SEAS.UPENN.EDU
Paul Lorenc PLORENC@SEAS.UPENN.EDU

Abstract

Jazz harmonization is the process of providing backing chords to a given musical melody. In this project, we set out to build a neural network to harmonize a given musical melody, using learned jazz motifs. We parsed and cleaned 125 jazz songs from the Weimar Jazz Dataset (Pfleiderer et al., 2017) and used these songs to train both CNN and compound CNN and RNN networks. Our dataset contained 186 unique chords, and we were able to accurately predict the correct chord about 7% of the time. However, deeper analysis shows that our model learns traditional musical ideas like tonal centers and jazz cadences. TyNet's ability to learn the key of the melody and harmonize accordingly, and embellishing with non-diatonic chords in a traditional jazz method, makes our model a useful education tool for composing and analyzing jazz harmony.

1. Introduction

This work comes as a response to a Google Magenta project that attempted to harmonize music in a classical method using a LSTM Neural Network train on the corpus of Bach's lifetime works (Huang et al., 2017). We set out to recreate a similar solution to harmonize melodies using a jazz method. The main challenge for this problem is the fact that jazz is much less formulaic than classical music, which by design makes it a harder problem to solve using pattern recognition.

In addition to this Google Magenta project, there was a recent attempt at jazz harmonization using SVC's from students taking an undergrad machine learning course at Stanford (Bien & Valdez, 2017). We hope to improve one their attempt by using neural networks, and not normalizing input melodies to the key of C, in order to preserve jazz, non-diatonic modulations.

This project aims to contribute to the current understanding of jazz harmonic analysis and become useful as an educational tool for jazz musicians.

University of Pennsylvania, CIS 419/519 Course Project.

2. Data

2.1. Data Source

We took the data from the public SQL Weimar Jazz Database (Pfleiderer et al., 2017), analyzed the database structure from the descriptions of different tables and columns. It exported columns relevant to notes and those relevant to the chords into two CSV files, using SQL queries. The data files were uploaded to the shared Google Drive and important in Google Colab using Python functions.

2.2. Data Formatting

Cleaning and reformatting the data was the first major hurdle for our project. There were no existing frameworks to parse and transform music data into standardized vectors that can be fed to machine learning models, so we had to build up a system ourselves. The first step we took was to convert the melody data found in the Weimar Jazz Database and convert it to midi files. This allows us to use the python library music21 (Cuthbert, 2020) to do further processing. We then took these midi files, performed quantization, and then sampled from them.

Quantization means that we align all notes to start exactly on beat. This removes any small errors and timing imperfections that were made in recording the midi data. Our dataset initially had very accurate timing data (down to fractions of a millisecond) from the jazz musicians who performed the original piece. Because they were human they were often a little fast or slow with regard to the beat. This makes it nearly impossible to programatically process the date, making quantization a necessity. However, it is important to note that playing slightly off time is often used in jazz to add "swing" or "feel" to a song, so we did remove this dimension of rhythm from our dataset.

Sampling means we record the midi data at a certain level of granularity. Like how pixels in a photosensor "sample" the color and luminosity of light in that area of the sensor and convert it to an array of integers, we sampled the notes

being played at each discrete moment in time. Sometimes the data set contains flurries of notes where a soloist plays 10 notes in one beat of the song. Other times the musicians will play one note for 4 straight beats. In order to standardize the amount of notes for every beat we record the current note being played 8 times per beat. We record notes as integers between 0-11, one integer for each note. Also if no note is being played we label this period of time with a -1.

After quantization and sampling we are left with 125 songs. These songs all vary in key but we made the decision of not normalizing the key of these songs unlike other previous jazz harmonization projects. This is because many key features of jazz come from purposely using out-of-key chords to complement diatonic (in-key) chords. By not normalizing the key of these songs we retain these features in our dataset, in an attempt for our model to also learn these features.
Each song is represented by a with a 1D melody vector containing the notes played over time, and a 1D chord vector containing the corresponding chord played at that moment in time.

2.3. Preprocessing

Because we did not normalize the chord and melody information to one key, we wanted to develop ways to make it easier for the model to find and distinguish areas where certain keys work better over other keys. We did this by adding extra channels to the 1D melody. Similar to how color images in a 2D convolutional network have 3 channels for each RGB distinction, we created channels that are the same dimension as the original melody, and each channel represents a chord. A given channel is "high" when the current note at that point in time is a part of the chord this channel represents.

C,	C,	C,	C,	C,	C,	C,	C,	C,	C,	C,	C,	C,	C,	F,	F,	F,	F,	F,	F,	F,	F,	F,	F,	F,	F
1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0

In this example we show the original melody (the top vector), and the corresponding C major channel (the bottom vector). A C major chord is constructed out of the notes C E and G, so when the note C is being played in the melody, the corresponding channel value is 1. When F is being played, which is not in the C major chord, the channel is low. One of the foundations of harmonization is to play chords that are contained in the melody, these channels will give the model 1D "edges" to distinguish when certain chords "work" with the melody. This information also makes it simple to distinguish the general key to play in, because in a given key, only 7 chords are used.

3. Methodology

3.1. CNN and Cascading Networks

We extracted 186 chords present in our test data and turned them into numerical labels for our CNN model. Using the pre-processed data, we were able to tune a neural network relying on temporal convolutions to predict chords surprisingly well for a given melody. The accuracy of the model depends on the nature of the song (features such as the key of the song) because jazz is often unpredictable, therefore, the exact prediction of chords is not always possible. However, even with these obstacles, our CNN model was able to predict chords with 7% accuracy on 10 random unseen songs.
Nevertheless, we found out that accuracy was not the best measurement to assess the performance, because, with a too complex model, our CNN was always converging into predicting the most popular chord, and get pretty good accuracy as for 186 different labels. Therefore, we also had to take other ways of measurements, such as ensuring the diversity of predicted chords while tuning our CNN model.
Additionally, we tried to implement cascading CNN where we would be predicting some X amount of the most popular chords in the "first-layer CNN model" or "OTHER" for a chord that is not in the top X most popular chords, and in a case when "OTHER" is predicted, proceeding to a "second-layer CNN" trained on less popular chords, in order to ensure that the less popular chords stand a chance of being predicted when there are many of them but with a small probability each.
Unfortunately, we were not able to find any value of X for which the first-layer CNN model wouldn't converge to always predicting "OTHER". For our model, regardless of its complexity, predicting "OTHER" was always a safer choice than predicting one specific chord. It turned out that just having one CNN with all 186 chords as labels were much better, after tuning it, it was able to predict the top 20 most popular chords in their correct places, and sometimes even predicting chords from the top 50. Other chords were so rare that they got basically ignored by the model, however, because of their rarity, they were not that significant for the quality of the produced music.
The parameters for which our model turned out to be the most optimal were using CrossEntropyLoss from PyTorch for the loss function with Adam with learning rate 0.0004 as the optimizer, batch size of 64 with 25 channels running for 50 epochs. Our model consists of a 1-D convolution layer with a kernel of size 8 and stride 8, batch norm 96 in 1D, ReLU, max pool with a kernel of size 4 in 1-D, dropout, a linear layer of size 384, and ReLU.

Table 1. Accuracy Results

MODEL	ACCURACY
BASELINE	6.02%
CNN	7.03%
COMPOUND	5.68%

Table 2. Distribution Results

CHORD	DIST. ORIGINAL	DIST. MODEL
BB7	6.02%	8.58%
F7	4.01%	4.19%
C7	3.09%	3.21%
EB7	3.01%	2.08%
D-7	2.03%	2.81%

3.2. RNN

An important aspect to musical composition is the recurrent reemergence of certain themes. After obtaining good results from our initial CNN we were wondering if there was any way to allow it develop some kind of memory of previously predicted chord patterns. We looked into implementing the RNN made available by PyTorch. The input was the on-hot-encoded output of our CNN. This was then tested against the correct chord labels.

After extensive hyperparameter tuning, we found the new chord predictions to consistently attain lower accuracy than those predicted by the CNN. While chord predictions did start to follow a prediction pattern, the patterns seemed to span at most 4 beats before repeating. This added an undesired monotonicity to our chord predictions and drove the overall accuracy lower. With an optimal parameter tuning the RNN achieved an accuracy 20% lower than the CNN. This is likely to be attributed to overfitting a given chord pattern to most of the inputs. Our model consisted only of 1 RNN layer and hence we could not reduce its complexity further. An alternative to the approach we could have taken could have skipped the CNN and fed the melody input straight into the RNN. Since this would no longer rely on a compound network, it might have brought about less overfitting in its predictions.

4. Results

In order to test for accuracy we created a validation set of 10 songs played in different musical keys. The most popular chord played was Bb7. It was played 6.02% of the time, hence we take this value to be the baseline which a trivial algorithm always predicting Bb7 would achieve.

Table 3. Art Pepper Anthropology in Bb Major

CHORD	ROLE IN THE KEY OF BB	FREQUENCY IN MODEL
G-7	RELATIVE MINOR	43.70%
BB7	ROOT	36.95%
AB7	NON-DIATONIC MEDIANT	6.74%
C7	II MAJOR	6.45%
EB-7	MINOR IV CHORD	1.76%

4.1. Analysis

Our CNN implementation consistently beat the baseline by a percentage point. Yet, due to the musical conventions of Jazz, accuracy isn't a valuation metric that is entirely capable of capturing the spirit of the genre. Performers are not meant to play in accordance with a given rule set, but rather are expected to rely on musical intuition in order to bring a piece to life in a new way.

For a given melody, there are an infinite way to "correctly" harmonize the notes using jazz theory. However, in jazz, chords are not created equal, the V-I cadence also known as the perfect cadence is a staple of jazz theory so in the key of Bb for example, we should expect to see both the V chord of the key (F) and the I chord of the key (Bb) more than the other chords in the key. Following this, the distributions of chords can give us insight into if our model is picking up on these jazz tendencies. As shown in Table 2, our model contains a similar distribution of chords to the original dataset, which tells us that our model is often finding the correct keys for songs and is also gravitating to the correct cadences at a similar frequency as the expert jazz pianists.

In Table 3 we have analysis for one of our test songs, Anthropology by Art Pepper. The model guessed mostly root, and root-related chords. These are relatively "safe" chords, they all feel resolved and will sound harmonious with the melodies which are all in the Bb major scale. There is a great lack of F7 chords in the guesses for this song, which could be a byproduct of how the melodies were formed. The model also generated a few interesting substitutions. The model backs 6% of melodies with the Ab7 chord, this same out-of-key chord is used in the actual chord progression for the song where it is used for about 5% of the beats. One reason our model consistently chooses "safe" chords is because there is no penalty for chord repetition. In jazz, sometimes chords are held for up to 8 measures continuously as the lead solos. However, in most situations unsafe, "tension" chords are brought up to add color to the progression, with the intention of resolving this tension later. Our RNN attempted to take into account past chords however the same problem persisted here as well.

5. Discussion and Conclusions

In conclusion we have built a model that generates novel jazz harmony for a given melody. We have learned a lot along the way, both about jazz and machine learning. Yet, we believe there is much left to solve in this problem. Our model correctly identifies and uses chords in the key of our test songs, and is even able to find non-diatonic chords that were part of the original song's chord progression.

However, the model primarily uses about 20 chords and only occasionally wavers outside of this top 20, never using diminished chords or chord extensions due to how the model was trained. Chord names themselves present a unique challenge because especially in jazz every chord has a handful of functionally identical chords. For example any diminished chord has the same exact set of notes as 3 other diminished chords. This means there are essentially only 4 different diminished chords, but in our model there are 12 due to the chord naming system we use to label everything. Another example is how a C major 6 chord is identical to an A minor chord. A note-based approach could help resolve this ambiguity.

One other area we did not cover was data augmentation. Any melody that in a given beat, can have their order and timing info changed in any way as long as it stays in the same beat. This can exponentially grow the amount of melodies one has to train with.

Finally, we have built a function to allow users to input arbitrary midi melodies and outputs the corresponding midi backing track to the melody. This could possibly be expanded to allow users to grade the harmony and input revisions. This user feedback, along with the aforementioned data augmentations, could be used to improve the model.

Acknowledgments

We would like to thank Dr. Dinesh Jayaraman for consistently providing us with insightful advice about possible neural network architectures we could explore in order to optimize the function of our model. This included cascading and recurrent neural networks.

References

Bien, N. and Valdez, L. Generating groove: Predicting jazz harmonization. 2017.

Cuthbert, Michael Scott. Music21, 2020. URL https://web.mit.edu/music21/.

Huang, Cheng-Zhi Anna, Cooijmans, Tim, Roberts, Adam, Courville, Aaron, and Eck, Douglas. Counterpoint by convolution. In *International Society for Music Information Retrieval (ISMIR)*, 2017.

Pfleiderer, Martin, Frieler, Klaus, Abeßer, Jakob, Zaddach, Wolf-Georg, and Burkhart, Benjamin (eds.). *Inside the Jazzomat - New Perspectives for Jazz Research*. Schott Campus, 2017.