Chorales, Chords, and Counterpoint: Generating Harmonizations from User-Given Melodies

Graham Lazorchak
Oberlin College
glazorch@oberlin.edu

Jacob Richter
Oberlin College
jrichter@oberlin.edu

Veronica Ayars
Oberlin College
vayars@oberlin.edu

Izzy Snyder
Oberlin College
isnyder@oberlin.edu

ABSTRACT

The aim of this project was to use machine learning to generate counterpoint following the style of Bach's chorales, inspired by the Bach Counterpoint Creator Google Doodle [1]. This is accomplished by feeding a melody line into a Markov model that generates chords based on sequences of notes and chords found in Bach chorales. This melody line and chord sequence is then given as input to a Convolutional Neural Network, which generates accompanying alto, tenor, and bass lines that harmonize with the original melodic line. This approach produced promising results.

1. INTRODUCTION

The application of computer science to art is a heavily explored and exciting area of inquiry. While art and photography are at the forefront, with creations such as Craiyon [2] and Lensa [3] generating a lot of media attention recently, music can also be merged with computer science. In this project, we explore how music can be interpreted by a machine learning algorithm, with the hope that sufficiently sophisticated models may be able to pick up on composition patterns and music theory rules that human composers draw on in their work.

The Bach Counterpoint Creator Google Doodle's approach is far from perfect; it can be fun for some quick experimentation, but the harmonizations tend to be rather boring and don't quite follow European music theory. One major source of these problems is the Google Doodle's data set. The Google Doodle is trained on a relatively small data set of 306 Bach chorales [1]. Additionally, its machine learning model only uses TensorFlow.js to allow all of its computations to run online instead of the user's computer, decreasing the possible computing power [1].

This moderately successful model presents a fun challenge: creating a machine learning model that produces better results. In the pursuit of more beautiful counterpoint harmonies, we developed a two-step solution: first, a Markov model harmonizes, or assigns chords to, a melody. Then, a Convolutional Neural Network (CNN) generates three musical lines based on the melody and chords to create the counterpoint. We were able to quadruple the size of the data set for our Markov model by dividing each Bach chorale into four separate training files, one with each voice (soprano, alto, tenor, and bass) as the melody. Additionally, we implemented a bash script to run both models sequentially, and a graphical user interface that allows the user to input a melody and display the finished counterpoint.

For our harmonization model, we developed two Markov models: a variation on a Markov chain and a Hidden Markov Model. Both Markov models yielded relatively low accuracy rates (around 42% to 50%). However, these were statistically significantly higher than a baseline of always guessing a C major chord (which produced accuracies between 28% to 35%). This is also impressive considering how many possible chord choices there are, and that many may be acceptable even if they are not the chords Bach wrote. The Hidden Markov Model statistically significantly outperformed the Markov chain, informing our decision to use the Hidden Markov Model in our two-step solution.

Our counterpoint model produced very impressive results using a CNN. However, it is difficult to evaluate our counterpoint model from an objective standpoint. If we simply compare the results of the network to Bach's solution, it isn't a meaningful comparison. A melody can be harmonized with contrapuntal lines in multiple valid ways, even with the same chord progression. However, we can state confidently that, simply by ear, the results of the CNN tend to sound pleasant, and that the results improved

qualitatively over the course of the training epochs. The model was trained using categorical cross-entropy as its loss function, and the loss decreased greatly over the course of the training epochs as well.

2. RELATED WORK

In addition to the previously mentioned Google Doodle, much research has been done to generate music and predict chord sequences. In this section, we present an overview of this area of research as a whole as well as specific implementations, such as Coconet and ChordAL, from which we drew inspiration for our approach.

The paper *Music Composition with Deep Learning: a Review* [4] discusses the methods used to generate music, the challenges of choosing good evaluation metrics for the mostly subjective task of making music, and questions of authorship and creativity. According to the paper, commonly used neural network architectures for music generation are variational autoencoders (VAEs), generative adversarial networks (GANs), and transformers. Common elements that all of these have is two independent pieces of the structure (encoder and decoder, generator and discriminator) passing the prediction back and forth to generate a result or improve at generating a result.

The paper mentions many specific projects which we looked at to better understand approaching this problem: Google's Magenta Melody RNN; Anticipation-RNN; DeepBach; MusicVAE, which is now one of the best models at creating a short motif; MusicTransformer; MuseNet; TransformerVAE; PianoTree; MICA and MSMICA for multi-instrument orchestration; Coconet; ChordAL for generating chord structures which inform note generation; and BebopNet for jazz. The author mentions that newer models use a combination of models to overcome problems like length constraints and randomness in note choice, and that some models use external parameters like self-similarity constraints to give the music more repetitious structure.

A challenge this paper brings up is evaluating ML/AI-generated music. The author claims that it is important to evaluate outputs subjectively by someone with a musical background, similar to how human composed pieces are evaluated. Objective measures are also mentioned, which include loss, perplexity, BLEU score, precision, and recall or F-score. There are also metrics related to pitch, rhythm, harmony that are useful for specifically evaluating music.

This paper presents many examples of models that are successful at some aspect of music generation, links to helpful datasets, and brings up important questions which are relevant to how we evaluated our model's performance.

Two models that have been particularly influential on our research are Coconet and ChordAL.

A few months after the Google Doodle was made public on Google's search page, the creators submitted a paper [5] detailing their techniques. The Google Doodle's inner mechanics are based on a re-implementation of Coconet [1], a model described in an earlier paper [6] by some of the same researchers. Coconet works in part by using a CNN, which takes a partially-masked section of music as input in the form of a piano roll, together with the shape of the mask. The network learns through stochastic gradient descent to replicate the masked region.

The distributions generated by this network are used with a Neural Autoregressive Distribution Estimator (NADE) [7] to predict the contents of a given piano roll. This by itself is typically not very clean, so the researchers used blocked Gibbs sampling to get a more pleasing harmonization.

The Google Doodle made a few alterations from the original Coconet implementation. Notably, they quantized the weights of the model, which led to reduced download size, but also potentially poorer performance—it is our opinion that the Coconet demo is more adept at producing harmonizations than the Google Doodle.

ChordAL is a model that generates music with an emphasis on chord structure and harmony. It is discussed in the paper *ChordAL: A Chord-Based Approach for Music Generation using Bi-LSTMs* [8]. ChordAL is made up of three parts: a chord generator, a chord-to-note generator, and a styler.

The chord generator uses a starting seed to generate a chord progression, which it has learned from a chord database. In the learning phase, chord vectors were passed into a 32-dimensional embedding layer. These embeddings were then passed into a stacked Bi-LSTM with 2 hidden layers of 64 neurons each. This method is based on ideas from natural language processing, which is similar in many ways to music processing.

The note generator outputs notes off of the chord output of the chord generator. To generate notes, chords are converted to their learned embeddings and then fed into a stacked Bi-LSTM with 2 layers of 64 hidden neurons each. The styling of the piece post-processes the output to make it more polished and listenable by removing random short notes and applying user-set instrumentation settings.

The paper claims that ChordAL learned chord relationships similar to the circle of fifths for determining chord similarity, and could therefore understand some fundamental musical relationships about harmony. Music evaluators who were surveyed on the harmonization, rhythm, and structure of ChordAL generated music rated the pieces very highly for harmony and moderately to low for rhythm and structure. Improving the rhythm and structure is an area of further research for the developers.

The datasets used in this model were the Nottingham dataset [9], The McGill-Billboard Chord Annotations dataset [10], and the CSV Leadsheet database [11]. Some of these datasets are publicly available on GitHub [12] due to ChordAL being open source.

3. PROBLEM

For our project, we aimed to improve upon the counterpoint generated by the Bach Counterpoint Creator Google Doodle. Similar to the Google Doodle, we wanted to allow a user to input a melody on a sixteenth note grid and have our model generate alto, tenor, and bass lines to form counterpoint with the user's melody.

The Google Doodle is trained on 306 Bach chorales from the JSB Chorales dataset [13] and uses TensorFlow.js to run the model entirely in the browser [1]. Its model is based on Coconet, a machine learning model created by Anna Huang, an AI resident of the Magenta Team [5] [6]. The biggest problem with the Google Doodle is the small size of its data set, which results in the occasional creation of a sub-par counterpoint. Our goal was to develop a better counterpoint creator through investigating different machine learning models, using a larger training data set, and using a two-step approach of first harmonizing a melody and then generating counterpoint based on the result of the harmonization, as this is similar to the workflow of many composers. The harmonization model would learn to assign chords for the user's melody at a given harmonic rhythm (for instance, every eighth note or every quarter note), and the counterpoint model would learn to compose counterpoint based on those chords and the melody.

4. SOLUTION

We created a model that operates in two parts: first, a Markov model takes in a melody line and produces a sequence of chords. Then, a CNN generates the final counterpoint. All of this relies on both functional models and a large dataset. Below we discuss the way the data was collected and preprocessed, as well as the specifics of our model implementations.

The Data

We used the same dataset used by Coconet and the Google Doodle, JSB Chorales [13], which encodes 382 4-voice Bach chorales as 2D arrays of integers. The data set divides the piece into 16th note chunks; each row of the 2D array contains the MIDI values of the notes for each of the four voices in one of the 16th note chunks. The main obstacle with this data set was that it did not contain chord data, but using a deterministic algorithm, we were able to assign chords to each time step of the counterpoint sequence and identify the key signature of each chorale. Additionally, we split each chorale into four files, each consisting of one of the lines (soprano, alto, tenor, or bass) along with the chord sequence and key signature, yielding 1528 individual melodies with chords for our harmonization model to train and test on.

For each model, we used 80% of our data for training and 20% of our data for testing. Since the four melody-chord files generated from each chorale have the same chords, we did not split up data generated from the same chorale between training and testing for the harmonization model.

For the harmonization model's training and test data, we normalized the notes to be between 0-11 because harmonic progressions are generally not dependent on a melody's octave. We also transposed all melodies into C major or its relative minor, A minor, because harmonic progressions are key-independent. All major keys have the same harmonic progressions and all minor keys have the same harmonic progressions.

The Models

Similar to ChordAL [8], our solution uses multiple stages. First, a Markov model determines how to harmonize the given melody by generating chords in the melody's key. Second, a CNN and sampling procedure similar to Coconet's uses those chords to produce a four-part harmonization. Splitting this process into two steps allows us to parallelize our work and fine-tune our models to be good at each task.

For harmonization, we developed two Markov models, the first being a variation on a relatively simple Markov chain. Instead of using only the n-1th chord to predict the nth chord like a typical Markov chain would, our model uses the n-1th melody note, n-1th chord, and nth melody note to predict the nth chord. Also, instead of randomly selecting the nth chord based on the probabilities of all the possible choices, we always select the chord with the highest probability. We also trained a model to predict the nth chord based on only the nth melody note, which we use to predict the first chord and any note/chord combinations that the model cannot account for because it does not exist within the training set. If this model also fails, we assign the nth chord to be the starting chord. In hopes of improving accuracy, we tried a more complicated model setup where we additionally considered the chord's location within the melody, but this did not significantly improve our results.

Our second harmonization model is a Hidden Markov Model, which we created using hmmlearn [14]. A Hidden Markov Model attempts to find the most likely sequence of states for a Markov chain given a variable observed at each timestep. In our case, the states are chords and we observe the melody at each timestep (specified as quarter or eighth notes); we want to find the most probable chord progression to harmonize the melody. Hidden Markov Models consider both transition probabilities, or the probability each state (chord) will follow the previous one, and emission probabilities. An emission probability is the percent chance that the observed variable (melody) will take on a specific value (note name) given a particular state (chord). Since the first state does not follow any other state, we also need a set of initial probabilities—how likely each chord is to be the first chord. Our program calculates all three types of probabilities by counting instances in the training set and stores the transition and emission probabilities in 2D matrices and the initial probabilities in an array. We also experimented with creating two separate Hidden Markov Models, one for major key melodies and one for minor key melodies, but this did not result in a statistically significant difference in accuracy.

Both Markov models contain a hyperparameter, harmonic rhythm, which determines the interval at which it observes chords in the training set and generates them when harmonizing a melody. For example, an eighth note harmonic rhythm would mean the model generates a new chord every eighth note.

For creating counterpoint, we used a CNN. Convolutions take advantage of the spatial nature of musical data. Shifting a passage of music up or down in pitch affects the actual note names, but not the relationships among them. Similarly, shifting a passage left or right in time doesn't change how one note moves to the next. However, like with images, the relative position of one note to another is highly important.

Rather than inputting an entire chorale's worth of data, the network deals with 32 timesteps (i.e., 32 sixteenth notes) at a time. We first attempted to train the model by dividing the chorales from the dataset into non-overlapping samples of 32 timesteps each. However, we found that we could increase the amount of data at our disposal by using overlapping samples with a stride of 4 timesteps. This was chosen because it maintains the placement of strong beats by shifting each sample by a quarter note.

The pitch range of the model was also constrained to the 46 notes present in the JSB Chorales dataset. While we did not artificially create more data by transposing chorales up or down from their original key, this may be an opportunity for future improvements to the training data.

We originally used a quasi-one-hot encoding scheme for the chord data. However, we found that the network learned better if we simply had 12 neurons in a column, each representing a different pitch class, and gave activations of 1 to the notes present in a given chord.

The network takes the melody and chords as separate inputs which are processed in parallel. The melody is fed into a series of convolution layers to learn spatial features, while the chords are fed into a series of dense layers. Eventually, the two layers are concatenated and fed through a single dense layer. From there, the melody creates a bass line. Finally, the bass line and dense layer are concatenated and densely connected to determine the alto and tenor lines.

The decision to generate the bass line first and then have it feed into the alto and tenor comes from the

fact that the bass is typically more important than the other two, since it greatly influences the quality of a chord.

When creating the output chorale, we had originally planned to sample the probability distribution of each column in each output layer, similar to Coconet's sampling procedure. However, this proved to be extremely chaotic; the model didn't learn the probabilities with enough confidence for the sampling to produce desirable results. Instead, we opted to simply pick the highest probability in each column. This, too, led to some issues. First, the highest probability was occasionally the first neuron in the column, which corresponded to C2, the lowest note in the pitch range of the data. This happened even when the note C wasn't present in the current chord, and happened in voices for whom that note had never appeared in the dataset. Upon examining the probability outputs, we found that this occurred when all probabilities in a given timestep were 0. This is likely a result of using ReLU activation, which sets all negative activations to 0. Second, the voice parts were extremely chaotic, often moving unidiomatically around their ranges. The bass voice in particular would often leap up an octave on one sixteenth note and then down an octave on the next.

To counter these issues, each unfiltered output layer is also followed by a two convolution layers, each with a single 3×5 filter.

The hope was that these filters would learn something akin to a Gaussian kernel, smoothing out the lines without the need for probabilistic sampling.

User Experience Components

In addition to creating two models, we also created a GUI called PillyLond for users to create their own melody, play back the melody, and play back the generated harmonies. The GUI includes different colors for each of the four voices, as well as MIDI playback of the chorale. Most of these behaviors are based on the Coconet editor. We did slightly change the way that notes are erased; in addition to the eraser tool, clicking on an active note with the same voice tool will deactivate it, and clicking on it with a different voice tool will replace it. This does mean that users cannot write unisons between voices, but it is far more intuitive for editing. When reading in a chorale, PillyLond prioritizes the color of the lower voice on a unison.

To allow the user to easily pass a melody file through both models, we created a bash script that runs the harmonization model and the counterpoint model sequentially. The bash script takes the file name containing the melody as an argument and passes it to the harmonization model. Once the model outputs a file with the melodic line with its predicted chords, the file is passed to the counterpoint model, which produces the final harmonization. The user can then load the harmonization into PillyLond and view the models' joint results.

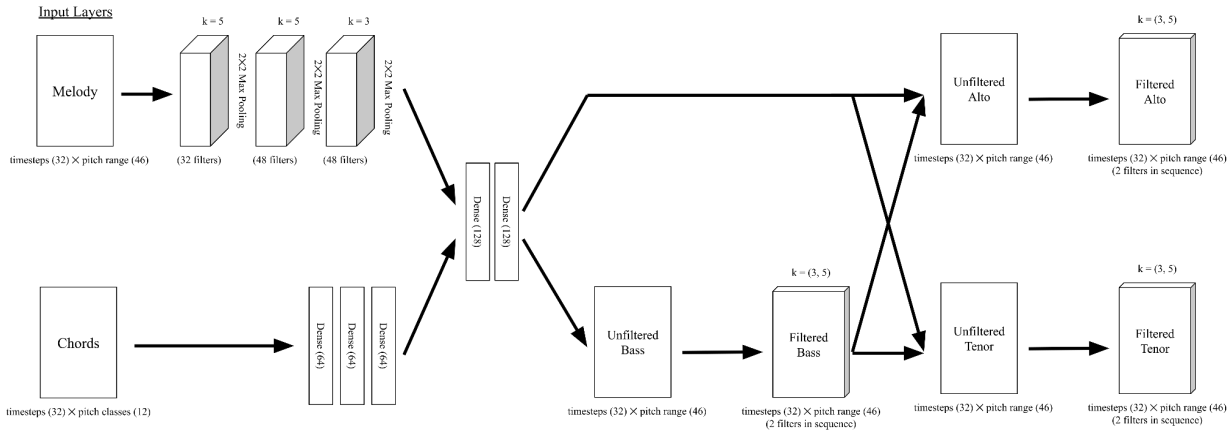

Fig. 1. Convolutional Neural Network structure.

5. RESULTS

Harmonization Models

Model	1/8 note harmonic rhythm			1/4 note harmonic rhythm		
	Accuracy	CI lower	CI higher	Accuracy	CI lower	CI higher
Hidden Markov	0.446	0.441	0.451	0.496	0.489	0.503
Markov Chain	0.421	0.416	0.426	0.450	0.442	0.457
Baseline (always guessing C major)	0.289	0.284	0.293	0.344	0.337	0.351

Table 1: Markov model accuracies. CI stands for 95% confidence interval.

To decide which Markov model to use for our harmonization model, we calculated each Markov model's accuracy chord by chord. We awarded the model full credit if the model predicted all components of the chord correctly, and we gave the model 2/3 credit if it predicted the chord's root correctly. If the true value of a chord was unknown (i.e., the deterministic algorithm failed to identify it), we did not factor the model's prediction of that chord into its accuracy.

We chose always guessing C major as a baseline for comparison because C major is the most common chord in the chorales and random guessing would be much less accurate. For both harmonic rhythms, the Hidden Markov Model and Markov chain model both had higher accuracies (between 42% and 50%) than the baseline (28.9% and 34.4%). Each model's confidence intervals do not overlap with the confidence intervals of the baseline, so we conclude that both models statistically significantly outperform always guessing C major.

While the accuracy of the Markov models may seem low, it is important to take into account that these models had a difficult task, as they had to choose between 106 chords that were present in the training data. Furthermore, the accuracy figures do not take into account the fact that there may be several acceptable chord choices for one musical interval; our accuracy is calculated only with respect to the chords Bach chose. Therefore, the accuracy measures seem better suited for inter-model comparison than an absolute evaluation of performance.

When comparing the Hidden Markov Model to the Markov chain model, the accuracy confidence intervals were non-overlapping for both note harmonic rhythms, indicating that the Hidden Markov Model is statistically significantly more accurate than the Markov chain model. One reason we believe this may be the case is if a certain combination of chords and melody notes never appears in the training set, the Markov chain has to resort to a less accurate model. The Hidden Markov Model, on the other hand, never needs to change approaches. Even when note-chord combinations are present in the data set, if they only occur rarely, the Markov chain model may experience overfitting. The Hidden Markov Model is more likely to avoid this because it consists of fewer, more generally applicable probabilities since the conditions for its conditional probabilities are less specific and thus more likely to occur often in the training chorales.

As for choosing a harmonic rhythm, we found that our Markov models were statistically significantly more accurate at generating chords every quarter note than every eighth note, perhaps because chord changes on offbeat eighth notes are comparatively rare and unpredictable to ones on the quarter note beat. We conclude that the most accurate harmonization model is the Hidden Markov Model with a quarter note harmonic rhythm. However, the eighth note Hidden Markov Model may be better if we want to generate more interesting harmonies; this is the model we used in our two-step approach. The Markov chain model is still usable despite its lower accuracy.

Counterpoint Model

It is difficult to evaluate the counterpoint model from an objective standpoint. We can compare the output of the network to Bach's solution, but this isn't necessarily a meaningful comparison; a melody can be harmonized with contrapuntal lines in multiple valid ways, even with the same chord progression.

The model was trained using categorical cross-entropy as its loss function, and we can plot the loss over the course of training, shown in Fig. 2.

We can also make more subjective evaluations on the model's performance. While the addition of filters on the voice parts did reduce some noise, the voices are noticeably more active than in the Bach examples. This could be reduced using ancestral or Gibbs sampling, but that would also require training the network on additional data with multiple voices, leading to increased training times. The lines also tend to be more chromatic and include dissonances uncharacteristic of the Baroque style. Some of these dissonances may be due to the final convolution filters pulling notes "off-course." While the voices are consonant with one another in most cases and tend to mostly use the notes present in the chord data, not all notes of the chord are always present in the final chorale. In particular, the third of the chord, which is one of the most important factors in determining a chord's quality, is often left out of the model's harmonizations. This was common in Renaissance counterpoint, but is practically unheard of in Bach's works. This missing third may be caused by treating the alto and tenor voices as independent outputs in the model architecture.

Utilizing our bash script, we were able to pass a melody to our harmonization model, pass the resulting chords to our counterpoint model, and generate three contrapuntal lines. While we did not have enough time to collect detailed metrics, we noted that counterpoint generated using both models tended to be more syncopated and dissonant than counterpoint generated by the counterpoint model alone, as shown in Fig. 3. This is likely due to the chords the counterpoint model is passed. When running just the counterpoint model, we used the original chords present in the Bach chorale.

Fig. 2. Loss over the course of training the CNN.

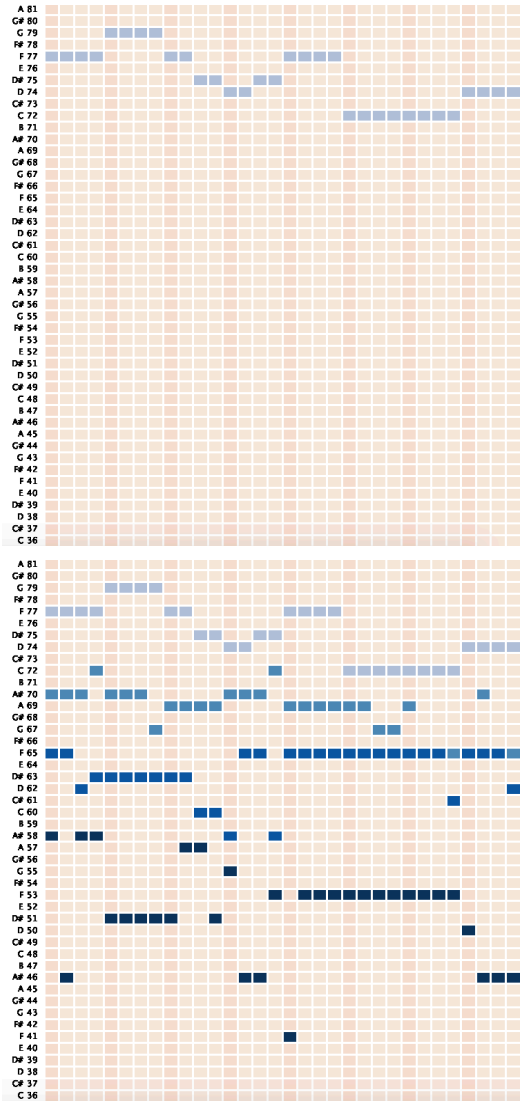

Fig. 3. The melody to chorale 358 (top), and a harmonization created using both the Hidden Markov and CNN models (bottom).

When running both models sequentially, however, we used the chords predicted by the harmonization model. This allows for more error in creating a counterpoint since both models can produce undesirable results.

6. CONCLUSION

Our project focuses on generating pleasing-sounding counterpoint based on chorales by J.S. Bach, with inspiration from the Bach Counterpoint Generator Google Doodle and its predecessor, Coconet. Unlike these two programs, our process involves two models. The first is a Markov model that harmonizes the user's melody, and the second is a CNN that creates alto, tenor, and bass lines given the user's melody and the chords generated by the first model. The resulting harmonizations do not match Bach's original chorales, but they do broadly follow Bach's counterpoint conventions and have interesting chord progressions. Additionally, the Markov models we tested were statistically significantly more accurate than a baseline of guessing C major for every chord.

Future implementations of this project could take several different avenues. Mechanically, we can work on connecting our bash script to our GUI so a user can export their melody, run it through both models, and import the resulting counterpoint without leaving the GUI. The models themselves could also continue to be tweaked and improved upon. There are still random notes and dissonances present in our current implementations, and the frequency of these could be reduced by training the CNN for more epochs or changing the parameters of our filters. Furthermore, expanding our training set with works from other composers in a wide range of genres could help the harmonization model become better at recognizing chord progressions present in other musical styles, like rock or jazz. Lastly, our results might improve if we used a GAN (generative adversarial network). Such a model includes a component that discriminates between human-composed chorales and the ones it generates, which could encourage it to achieve a more similar harmonization to human composers.

REFERENCES

[1] "Celebrating Johann Sebastian Bach," *Google*. [Online]. Available: https://www.google.com/doodles/celebrating-johann-sebastian-bach [Accessed: 28-Oct-2022].

[2] "Craiyon." [Online]. Available: https://www.craiyon.com/ [Accessed: 19-Dec-2022]

[3] Z Sottile, "What to know about Lensa, the AI portrait app all over social media, " CNN, 11-Dec-2022. [Online]. Available: https://www.cnn.com/style/article/lensa-ai-app-art-explainer-trnd/index.html. [Accessed: 19-Dec-2022]

[4] C. Hernandez-Olivan and J. R. Beltran, "Music composition with Deep Learning: A Review," arXiv.org, 27-Aug-2021. [Online]. Available: https://arxiv.org/abs/2108.12290v1. [Accessed: 28-Oct-2022].

[5] C.-Z. A. Huang, C. Hawthorne, A. Roberts, M. Dinculescu, J. Wexler, L. Hong, and J. Howcroft, "The Bach Doodle: Approachable music composition with machine learning at scale," arXiv.org, 14-Jul-2019. [Online]. Available: https://arxiv.org/abs/1907.06637. [Accessed: 28-Oct-2022].

[6] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," arXiv.org, 18-Mar-2019. [Online]. Available: https://arxiv.org/abs/1903.07227. [Accessed: 28-Oct-2022].

[7] H. Larochelle, and I. Murray, "The neural autoregressive distribution estimator." [Online]. Available: http://proceedings.mlr.press/v15/larochelle11a/larochelle11a.pdf. [Accessed: 28-Oct-2022].

[8] T. H. Hao, "Chordal: A chord-based approach for music generation using Bi-LSTMs." [Online]. Available: http://computationalcreativity.net/iccc2019/papers/iccc19-demo-9.pdf. [Accessed: 28-Oct-2022].

[9] E. Foxley, Nottingham Database. [Online]. Available: https://ifdo.ca/~seymour/nottingham/nottingham.html. [Accessed: 28-Oct-2022].

[10] "The McGill Billboard Project," The McGill Billboard Project · DDMAL. [Online]. Available: https://ddmal.music.mcgill.ca/research/The_McGill_Billboard_Project_(Chord_Analysis_Dataset)/. [Accessed: 28-Oct-2022].

[11] "CSV Leadsheet Database ," CSV_Leadsheet_DB. [Online]. Available: http://marg.snu.ac.kr/chord_generation/. [Accessed: 28-Oct-2022].

[12] gudgud96, "Gudgud96/Chordal: Code accompanying ICCC 2019 creative submission paper - 'Chordal: A chord-based approach for music generation using Bi-LSTMs'.," GitHub. [Online]. Available: https://github.com/gudgud96/ChordAL. [Accessed: 28-Oct-2022].

[13] Czhuang, "JSB-Chorales-Dataset," GitHub. [Online]. Available: https://github.com/czhuang/JSB-Chorales-dataset. [Accessed: 28-Oct-2022].

[14] hmmlearn developers, "HMMLEARN." hmmlearn, 2010. https://hmmlearn.readthedocs.io/en/latest/.