CS224N Project Report: From Note2Vec to Chord2Vec

Phil Chen

philhc@stanford.edu

Edward Xu

edxu24@stanford.edu

Mentor: Anand Dhoot

Abstract

Between music and natural language, individual musical notes are analogous to characters and chords analogous to words, but the distinction between notes and chords is much more blurred. Here, we use domain knowledge to create vector representations that express chords as sums of their constituent notes, thus enabling efficient and effective representations of musical chords. We evaluate these vector embeddings on simple music theory tasks and demonstrate that these embeddings capture fundamental properties that are meaningful in music theory. Future work may involve evaluating the effectiveness of these vector representations as components for existing music generation models.

1 Introduction

In this paper, we explore the generation of music through the use of embedding vectors to represent musical chords. Since chords in music can be compared to words in natural language, we drew inspiration from the word2vec and skipgram model to build our chord vectors [7]. Our goal is to generate embeddings that are able to represent basic music theory concepts, such as major or dominant chords. While there are billions of words to be embedded in language, there are only 88 musical notes, which makes it easier to extract relationships between notes from data. However, generation of music through these chord embeddings can be more difficult than its counterpart in natural language due to the small size of chords. Compared to lengthy sentences that we have in language, three or four note chords are not able to convey as much information as sentences.

To combat this problem, we utilize a sum representation of chords and also overtones to emphasize the key notes in the chord (fifths, octaves) which assist our model in realizing music theory concepts. After making these adjustments, our model is able to achieve a sufficient accuracy on classifying major/minor chords and root chord names.

Current methods to generate music are successful in utilizing a music transformer based on self-attention [3]. Although our approach does not involve a transformer, we believe that the chord2vec approach is nuanced in bringing out music theory concepts. While much of other similar work focuses on successfully generating music, our model puts more emphasis on the chord embeddings, which may be an essential stepping stone to learning music theory concepts and therefore generate more accurate music.

2 Related Work

We were primarily inspired to pursue a project with musical context after seeing the results of the Music Transformer [3]. The transformer's self-attention model and long-term memory is able to generate music almost identical to the original. Each layer of the model contains an attention sub-layer and a foward sub-layer, which is effective in bringing out the notes that matter most. We were thoroughly impressed by the seemingly real music generated by the transformer.

Papers like [1], [6], and [7] confirmed that learning chord embeddings, like word embeddings in word2vec, would be effective in representing musical chords and music theory concepts. From these papers, we were able to determine that the chord embeddings could potentially affect the musical orientation of generated notes and give more emphasis to musical concepts like the Circle of Fifths. From word2vec [7], we analyzed the skipgram model and took away from the idea of a context window. However, after closer examination, we concluded that word2vec could only give us a foundation, as chords had some significant differences with words.

To investigate further, we started to implement our note embeddings with a similar structure to the methods outlined in note2vec [1]. The note2vec paper utilized a skigram model with negative sampling to model slices of Beethoven sonatas. Unlike the word2vec paper, their model was optimized for small cosine distance between similar musical notes. Furthermore, their TSNE plots demonstrate a similar cosine distance among related notes, such as C and G.

Lastly, Music Generation with Deep Learning [4] gave us insight on how to use LSTMs to generate music. Their paper outlines a 1D convolutional architecture and even a bilinear LSTM approach where we use multiple LSTMs in a middle layer.

3 Approach

The general approach seeks to use ideas from word2vec [7], combined with music theory, to create a better representation system for musical notes. Our code can be found in this repository: https://github.com/philhchen/note2vec

3.1 Standard Note2Vec

The first approach is to use the ideas of Word2Vec and apply them directly to notes, similar to models in [1] and [6]. Our dataset contained series of chorales, which were each composed of series of chords containing 3 or 4 notes simultaneously. For each note in a chord, we applied SkipGram training using the note as the central "word" and all other notes of the chord as "context words." For example, if a chord contained the notes $(22, 28, 32, 34)$, this would provide the training data pairs $(\text{note, [context]}) : \{(22, [28, 32, 34]), (28, [22, 32, 34]), (32, [22, 28, 34]), (34, [22, 28, 32])\}$

We also included negative samples, with one negative sample per positive sample. Since the dataset contained a total of 52 different notes, we experimented with embedding dimensions of 4 and 8.

3.2 Note2Vec using Overtones

To better represent notes as vectors, we take advantage of the physics of overtones. When a note N is played on an instrument, overtones of notes $N + 12, N + 19, N + 24, N + 28, N + 31, N + 34, N + 36, \ldots$ can also be detected [8]. Thus, with our 52-dimensional vocabulary, we create 88-dimensional embeddings $(88 = 52 + 36)$, with each note N containing 8 nonzero entries in positions $N, N + 12, N + 19, N + 24, N + 28, N + 31, N + 34$, and $N + 36$, as shown in Figure 1.

3.3 Chord2Vec Superposition

While existing music generation models use serialized music representations that represent each chord as a series of disconnected notes ([5], [2]), musical chords are actually thought of as combinations of notes in the minds of listeners.

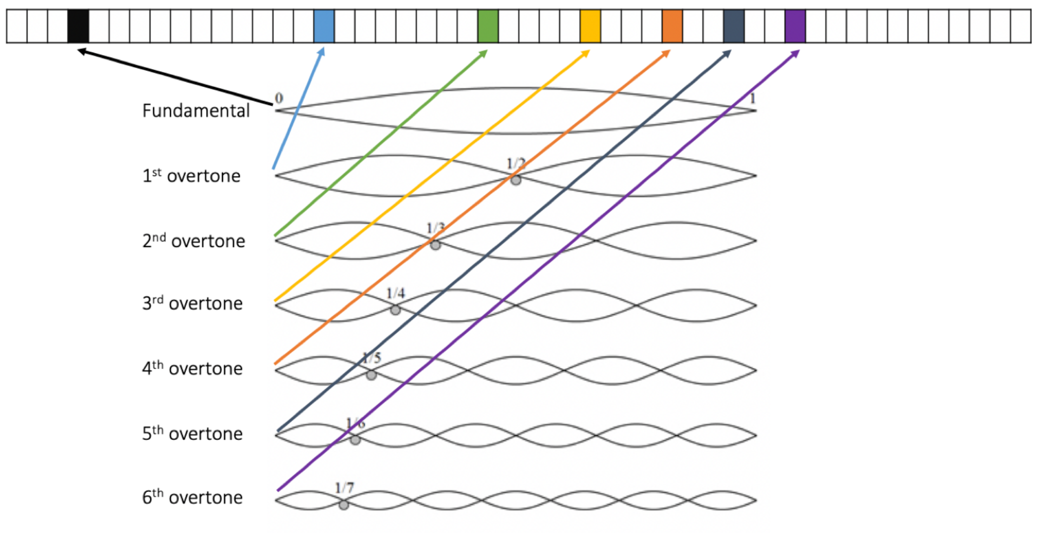

Figure 1: Note2Vec embeddings are created by mapping each note to a linear superposition of its fundamental frequency and its overtones

To represent chords, we consider each chord as superpositions of their constituent notes. For example, for the chord composed of the notes $23, 29, 31$, we represent the chord's vector as the sum of the vector representations of each of the notes.

3.4 Note2Vec Averaged

We trained our Note2Vec system using these 88-dimensional embeddings (with masks to restrict each note's nonzero dimensions to itself and its overtones). For an embedding function E and each chord $C = (C_1, C_2, C_3, C_4)$ where C_k are notes, we pass in the following as (word, context) pairs for skip-gram training: $\left(E(C_i), \frac{\sum_{j\neq i} E(C_j)}{3}\right)_{i=1,2,3,4}$. This represents each note's "context" as the average of the three "context notes" since overtones are theoretically additive in physics and sound [8].

3.5 Chord2Vec Training

To further train the Note2Vec based on similarities between adjacent chords, we adopted the paradigm of skip-gram training for chords with context windows of size 1. Formally, for any adjacent chords C_{n-1}, C_n, C_{n+1}, where each $C_k = (C_{k,1}, C_{k,2}, C_{k,3}, C_{k,4})$ is a chord and each $C_{k,l}$ is a note, we train pass in the following as (word, context) pairs for skip-gram training: $(\sum_{i=1} E(C_{k,i}), \sum_{i=1} E(C_{k-1,i}))$, $(\sum_{i=1} E(C_{k,i}), \sum_{i=1} E(C_{k+1,i}))$. This represents each chord C_k as the sum of its constituent notes, and the loss in skip-gram training backpropogates to embedding representations of individual notes, which will train the embeddings to better capture similarities between chords.

3.6 Chord Classification

Using our chord embeddings, we used a convolutional neural network to test our embeddings against a baseline model, which was a generic embedding of one-hot vectors. The goal was to first accurately classify a given chord (defined as a combination of three notes from the root note's scale) as major or minor, and then to classify a chord's root note from the 12 possible notes of the scale.

In the baseline one-hot vectors model, a 1 indicates a note in the chord and 0 indicates no note. The convolutional neural network includes a single layer of convolutions involving 64 channels each with kernel sizes of 13, 10, and 7, followed by a dense network of the convolutions to the number of

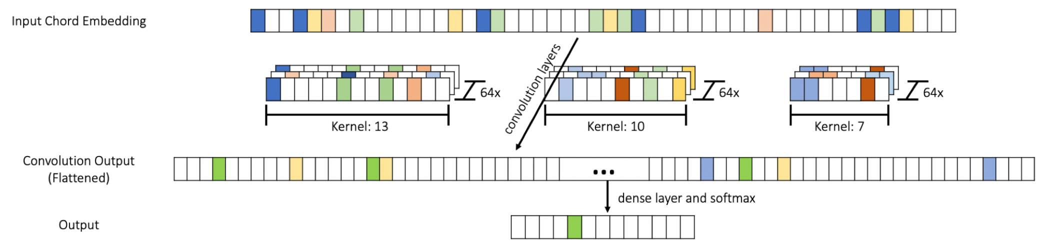

Figure 2: The neural network for chord classification involves a convolutional layer followed by a dense layer that outputs a prediction for the class of the chord (major/minor or root note).

output channels. For predicting major/minor chords, a global max-pooling operation is also included to identify the appearance of specific patterns in the chord embedding. A cartoon of the network is shown in Figure 2.

3.7 Music generation using LSTMs

We used the Chord2Vec embeddings to train a language model on a dataset of Bach Chorales using a bidirectional LSTM. Using this language model, we attempted to generate melody and harmony simultaneously, in contrast with previous work which has centered on generating music serially ([2], [5], [4], [3]). Various architectures involving attention were also attempted, but none of these models yielded any successful music generation results.

4 Experiments

4.1 Data

We are using the JSB-Chorales-Quarter dataset from https://github.com/czhuang/JSB-Chorales-dataset. This is a pkl file that includes the notes, represented as a number between 21 and 108, that are played simultaneously in each time step. There are a total of $23,134$ chords in the training set.

For chord classification, we generated 349 chords, each annotated with a root chord (expressed as a number between 0 and 11 inclusive) and whether it is major or minor (expressed as a boolean). We split the chords into 174 chords for the training set and 175 for the test set.

4.2 Evaluation Method

We use L1 loss of the standard skip-gram training algorithm as our loss function. For chord classification, we evaluate our model on its accuracy, which is calculated as the proportion of chords in the test set for which the model's prediction matches the actual class.

4.3 Experimental Details

For skipgram chord2vec training, we used a learning rate of 0.1, batch_size 32, 50 epochs, and standard SGD optimizer. For chord classification, we used a learning rate of 0.00001, batch_size 1, 50 epochs, and standard SGD with cross-entropy loss. For LSTM music generation, we used a learning rate of 0.01, batch_size 1, 50 epochs, and standard SGD with L2 loss.

4.4 Results

We see that using the overtone-based embeddings yields significantly lower training losses than the 4-dimensional and 8-dimensional embeddings, especially when considering each "context chord" as one entity (Figure 3).

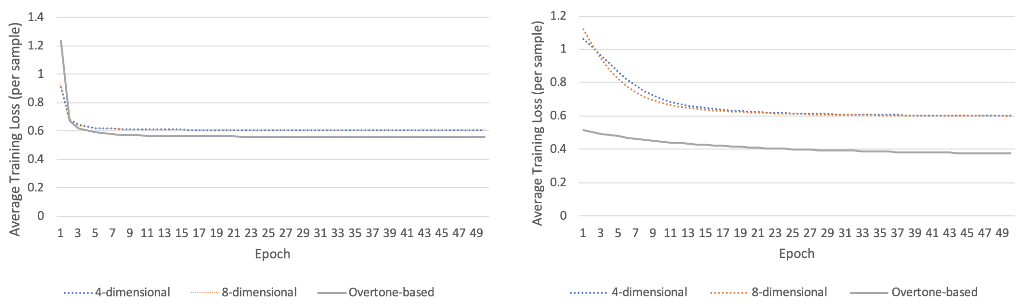

Figure 3: Training loss over epochs (average per sample) Left: results from standard skip-gram training approach (Section 3.1) Right: results from treating all "context notes" as a single chord (Section 3.4)

Using our chord embeddings, we tested our embeddings for their ability to "learn" music theoretic concepts by applying a neural network over the embeddings to classify chords as major or minor and to identify their root note. First, we fed the network major and minor chords of different inversions (root, first, and second) and trained it, then gave it the testing chords and recorded the accuracies, given below.

	Major/Minor	Root Note Name
Baseline Embeddings (Train Accuracy)	100%	100%
Baseline Embeddings (Test Accuracy)	81.1%	84%
Chord2Vec Embeddings (Train Accuracy)	100%	100%
Chord2Vec Embeddings (Test Accuracy)	100%	97.7%

Although we have high accuracies across the board for both baseline and chord embeddings, we can see that our Chord2Vec embeddings have a much higher test accuracy than the baseline vectors for both Major/minor chord classification and the root note name classification. This demonstrates that our embeddings are definitely able to learn the difference between chords of different qualities and note names.

5 Analysis

Our qualitative results consist of evaluating our Note2Vec embeddings. First, we implemented a baseline where for each focus note, we took each note in its chord and incremented its count. This resulted in finding the notes that occur the most with each focus note. To compare our embeddings to our baseline model, we plugged our embeddings into an embedding projector to visualize the closest notes to a focus note. The three "closest" notes for a few selected focus notes (C5, G4, D#3, F#5) are shown below.

	C5	G4	D#3	F#5
Baseline Counts	G5, C4, G4	G5, D5, C5	A#4, D#5, G4	D5, A5, D4
Note2Vec	C#5, D#3, G#5	G6, G2, D#4	C7, C#3, C#6	D#6, A3, A#3
Note2Vec with Overtones	C4, C6, G6	G5, G3, E4	D#4, F#3, C3	F#4, F#6, C#6

As we can see, the baseline method is successful in highlighting octaves and fifths, which have a prominent place in music theory. It is apparent that C5 includes G and C, D# includes A# and D#, etc. These notes are closest to the focus notes because they are also often seen in the chords surrounding the focus notes. On the other hand, the naive Note2Vec method is successful in catching consecutive notes and more effective in predicting which notes come after a particular note. For instance, in our Note2Vec, "C#5, C5" and "C#3, D#3" are consecutive pairs. However, because the naive Note2Vec model pays attention to a window of notes, it loses a lot of the relation to octaves and fifths. It is obvious that the naive Note2Vec method is ineffective in bringing out fifths and octaves and therefore does not pertain to a coherent music theory structure. We can conclude that there is a tradeoff between predicting notes that come next and retaining a strong musical relation within a context of notes.

However, in our approach of Note2Vec with Overtones, we were able to use overtones to give emphasis to fifths and octaves, so our closest notes to the focus notes paired extremely well with music theory concepts. For instance, C5 was closest to "C4, C6, and G6" and F#5 was closest to "F#4, F#6, and C#6", which are all octaves or fifths away from their focus note. Through projecting these embeddings and finding closest notes, we can conclude that the embeddings are effective in highlighting key notes in music.

In addition to these closest notes, we can see that the plot of our embeddings conveys a relation between root, third, and dominant (fifth) notes. To plot the embeddings, we transformed the embeddings to two dimensions using t-distributed stochastic neighbor embedding. This plot shows the relative cosine distance between the notes, similar to how words are plotted in word2vec.

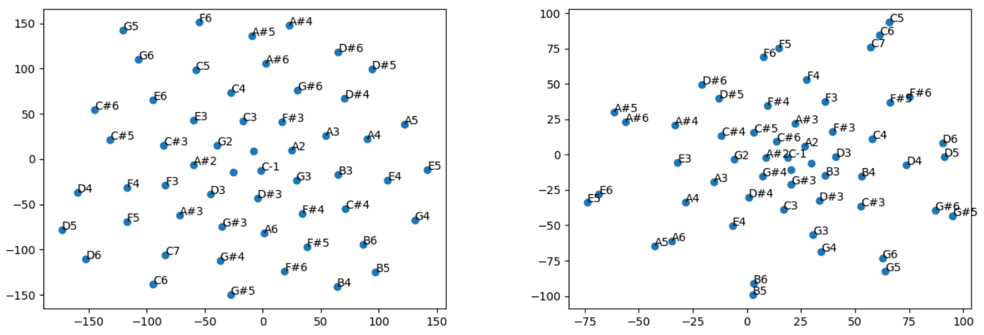

Figure 4: Note2Vec Overtones Embeddings TSNE Plot (Left: Perplexity 30, Right: Perplexity 2)

From the left plot, we can see that notes that are a fifth apart, such as E and A, have similar cosine distances. For instance, the vector from E4 to E5 is very similar to the that from A4 to A5. Similarly, the vector from C4 to G4 is consistent with the one from B5 to F#5. These similarities demonstrate that Note2Vec has learned basic music theory based on the Circle of Fifths, where the fifth note in a key is considered dominant. We can even see clusters of similar notes beginning to form, as all the notes names are close to each other, crowding around our padding note C-1 in predictable fashion.

The right plot, using a low perplexity, illustrates the clusters of our embeddings. We can see that the clusters are very predictable and generally match music theory concepts. For instance, all notes of similar note names cluster together (C, A#, F#, A). More importantly, we can clearly see separation of octaves in this plot. Notes of higher octaves surround notes of lower octaves, as all notes of octaves 5 and 6 are on the outskirts of the plot. Generally, we can see that the closer to the center (padding note), the lower the note's octave. This plot indicates that our embeddings have clearly delineated between notes and their octaves and therefore abide to certain musical concepts.

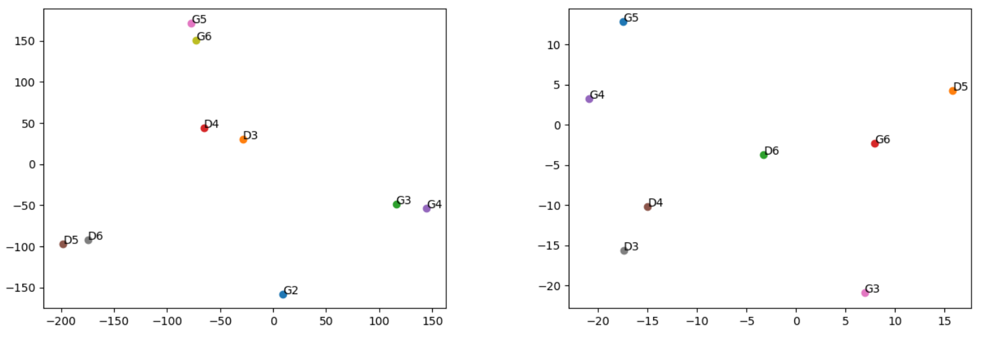

Figure 5: G and D Embeddings TSNE Plot (left: with Overtones, right: without Overtones)

Taking a closer look at why we use overtones, the two plots above take only two note embeddings from both regular Note2Vec and Note2Vec with Overtone models. Isolating just the G and D embeddings in our Note2Vec with Overtones, we can see that there is correlation between the two notes, especially of similar octaves. The vector from G3 to D3 is very similar to the one from G4 to D4. Moreover, the distance and vector between "G5, G6" and "D5, D6" are almost identical. This further supports the quality of our embeddings to derive musical concepts.

Comparing with the plots of G and D embeddings in the basic Note2Vec without overtones, we can see that cosine distances in the basic Note2Vec are not correlated. Instead, the placing of the octaves and notes in the naive Note2Vec seem very random. For instance, the vector between G4 and D4 does not match with the one between G3 and D3, and same for that of the fifth and sixth octaves. Note2Vec with Overtones is able to generally cluster all G and D's with each other, so we can see that overtones are crucial in bringing out important notes in music theory.

We also tried evaluating the closest chords to our chord embeddings, where the focus chords are C, g, d, and F. Uppercase indicates Major, while lowercase indicates minor.

	C	g	d#	F#
Baseline Counts	G, C, E	c, d, g	a, f#, d#	D, B, F#
Chord2Vec	C, a, G	c, g, A#	F#, d#, B	F#, a#, C#

The table illustrates that our Chord2Vec model has learned some basic music theory, as the closest chords to the root chords include some dominant chords (g and c, F# and C#). Furthermore, one of the closest chords to "C Major" is "a minor", and one of the closest chords to "d# minor" is "F# Major". These are the major and minor chords have the same key signature, and our chord embeddings were able to learn the similar key signature between the major and minor.

6 Conclusion

In this paper, we presented a novel vector representation system for notes and chords in music. Using physical and musical theory, we create vectors that can simultaneously represent notes and chords in music. We show that these representations capture fundamental properties of chords in the form of their root note and whether they are major or minor. Analysis of the nearest neighbors to individual notes shows the results correspond with what is expected from music theory.

7 Future Work

Because these vector representations capture much music theoretic information about notes and chords, we believe these representations will be useful in music generation. Though we were unsuccessful in generating music with LSTM models, we believe other types of deep learning architectures, like VAEs and GANs might benefit from the simplicity of the Chord2Vec representation. Future work could explore the possibility of applying Chord2Vec to the networks noted above.

References

[1] D. Herremans and C.-H. Chuan. Modeling musical context with word2vec. *arXiv preprint arXiv:1706.09088*, 2017.

[2] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck. Counterpoint by convolution. 2016.

[3] 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: Generating music with long-term structure. *https://arxiv.org/pdf/1809.04281.pdf*, 2018.

[4] V. Kalingeri and S. Grandhe. Music generation with deep learning. *arXiv preprint arXiv:1612.04928*, 2016.

[5] F. Liang. Bachbot: Automatic composition in the style of bach chorales. *University of Cambridge*, 8:19–48, 2016.

[6] S. Madjiheurem, L. Qu, and C. Walder. Chord2vec: Learning musical chord embeddings. In *Proceedings of the constructive machine learning workshop at 30th conference on neural information processing systems (NIPS'2016), Barcelona, Spain*, pages 1–5, 2016.

[7] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013.

[8] A. Wood. *The physics of music*. Read Books Ltd, 2013.