Harmonic Extension Using Deep Learning Networks

Eric Durbin [1]

Abstract

For experienced jazz musicians, the written harmony of songs is just the starting point in their musical journey, and creative players can easily find themselves venturing far outside the traditional structures of the written chord progressions in favor of more experimental harmonies. In live settings this experimentation can be easily matched by well-versed piano players who learn to recognize such spontaneous changes in harmony, but this experience is one that is sorely lacking in a solo practice setting. Fortunately, this type of structure is perfect for the domain of machine learning, and this project demonstrates that machine learning models can be used with a high degree of confidence to extend the harmony of composed melodies, as well as providing inspiration for creating new melodies over known chord progressions.

1. Introduction

While music appears to be the realm of pure creativity, under the surface there exists a surprising amount of rigid structure. This structure acts almost like the grammar of a language, and much in the same way that natural language processing can predict human text, music models have the potential to predict musical structures. A common process in the music composing world is to think of a melody and then come up with the harmony that best fits that melody. It may be thought that music is not the realm of algorithmic tools, but contrary to that belief, this is a process that could be improved by machine learning, as statistical models have the ability to find similarity in high-dimensional abstract spaces such as music.

This project aims to create a process for suggesting harmonic extensions of melodies by calculating the similarity between the melodic structure and chords of differing harmonic complexities. A secondary goal is to make the process performant enough to run in real time, so that, if combined with an audio transcription model, the two models can be used to create automatic real-time piano accompaniment for musicians.

1.1. Related Work

Work has been done recently expanding the field of audio and music processing. Genre classification between 5 different genres of music using various features of audio signals was achieved with a high degree of accuracy in (Chatterjee et al., 2024). This shows that learning is possible using non-transcribed sound signals over some problems. Additionally, it was found in (Silva and Turchet, 2024) that pattern recognition is possible using MIDI files, and was the inspiration for attempting to utilize an RNN for this project.

To connect live audio to MIDI for ease of processing, automatic music transcription was performed in (Edwards et al., 2023), as well as (Jamshidi et al., 2024). These techniques will could help connect projects like this one to non-transcribed performances - drastically increasing the data available for training.

1.2. Music Background

Music theory is an incredibly deep field and cannot be fully covered in this paper, however, some basics will be helpful to understand the later sections of this project. Generally, when songs are composed, both a melody and a chord progression are created. The melody serves as the main idea of the song while the chords function as the underlying structure giving the melody harmonic context. While these two parts are not completely independent, there is a lot of freedom when it comes to choosing which chord to apply to a given melody, as many chords share many of the same notes. Take for example G minor-7 and B$^\flat$ major. The G minor-7 chord contains the notes (G B$^\flat$ D F), while the B$^\flat$ major chord contains the notes (B$^\flat$ D F). Notice how even though one is major and one is minor, the major chord contains the entire minor chord within itself! This type of harmonic fluidity proved to be a challenge throughout the project, as many chords were enharmonic equivalents and thus had no differences for the models to learn.

Another key point is that the relationship between chords and melodies is bidirectional. In essence, while it is common to start with a melody and construct a chord, such as in

[1]Purdue University. Correspondence to: Eric Durbin <durbin3@purdue.edu>.

Statistical Machine Learning Project Report

composing, it is equally common to start with the chord and create a melody that matches. This latter part is commonly found in jazz improvisation, and is the primary motivation for this project. This means that there are two possible applications of the information gleaned from this project. One of them is to re-harmonize known melodies with different functioning but harmonically similar chords, and the other would be to take known chord progressions and play more harmonically complex scales over them. This second application is detailed further in Section 4, but an example that the model found would be with the seventh chord. The Seventh chord, also known as the Dominant chord, is the most common chord in jazz songs. It normally utilizes the Mixolydian scale (a very major-sounding scale), but an equally viable alternative would be the Whole-Half Diminished scale – one of the most dissonant scales possible! The reason for the extreme swap is that the Diminished scale contains a dominant chord within it, and thus will sound harmonically justified, even in songs that are not diminished in nature.

2. Dataset

The dataset used was a custom dataset of generated MIDI files, where each file contained a single chord or note sequence generated from a specific musical mode. The initial experiments, as well as the inspiration for the data generator used in later experiments, utilized the Jazznet (Adegbija, 2023) dataset. For training, 14 chord qualities were used, with each chord having 10 samples generated in each of the 12 keys. With a train/validation split of $80/20$ there were a total of $2,150$ training samples and 538 validation samples. For the test dataset, instead of sampling from the chords directly, scales with high similarity to the chords were used to generate mock melodies. This way of testing mimics applying the model over a real-world melody, as most melodies are composed of more than just chord tones. With 1 scale per key generated for each of the 14 chord qualities, a total of 168 test patterns were generated.

2.1. Data Generation

The dataset was generated using the idea that each chord and note pattern is independent of which musical key it resides, as well as the fact that each chord and scale can be constructed algorithmically using intervals between the notes. The C major chord, for example, consists of three notes: (C, E, G). Since major chords are always a root note, a major third, and a perfect fifth, we can represent the C major chord in interval notation as follows: (0, 4, 3), where each number is the number of half-steps above the previous note. Since the interval notation is independent of the starting note, 12 patterns can be generated for each chord quality by simply adding 1 to every interval in the chord for each desired key. Furthermore, noise was added to the durations of the notes to increase the variance of each training sample – a technique also utilized by (Jamshidi et al., 2024). Each note was added with a different duration sampled from a normal distribution with the mean duration centered at a quarter note (1 beat) and the standard deviation of a sixteenth note (0.25 beats).

In addition to each chord having a distinct set of notes, each chord has a family of scales that share common notes with the chord. For the test dataset, patterns were generated by sampling notes from these scales, with the majority of the notes programmed to be diatonic to the chord in question. For example, to generate a pattern of 50 notes from the Diminished chord with 70% being diatonic, 35 notes would be sampled from the diminished chord, and the remaining 15 notes would be sampled from the Whole-Half Diminished scale. Note that some of the notes in the Whole-Half Diminished scale are inherently diatonic to the chord, so the diatonic probability is actually around 85% in this example.

2.2. Data Representation

The process of constructing the input data involves loading MIDI files, parsing the internal tracks, and transforming the note events into a matrix of (occurrence, duration) pairs.

Since MIDI is a common format for music writing software, parsing a MIDI file is made trivial with various python packages such as Mido, Miditoolkit, and PrettyMIDI. Each MIDI file consists of note-on and note-off events, a number in the range of (0-127) representing the note to which the event corresponds, and a time counter for the time elapsed since the last event. The parser iterates over each event, remembering when each note on/off event took place, while adding up the total duration and the number of times the note was turned on.

The final data representation is a (12,2) matrix where each row corresponds to one of the 12 music notes, starting with C, and the columns were numbers between 0 and 1 that correspond to proportions for how many times that note appears and how long the note is on relative to other notes in each track.

2.3. Alternative Data Representations

Alternative data formats were attempted and scratched for various reasons. The initial attempt was to generate spectrograms from audio files, which resulted in a $(1024, t)$ sized image, where t is the number of steps sampled from the file. While the spectrogram format is theoretically a perfect mapping of audio data, it was found that the excess noise in the form of overtones, as well as the massively increased dimensionality of the input space, made learning over this

Table 1. Chord Qualities and Corresponding Scales Used.

NAME	CHORD SYMBOL	CHORD NOTES	SCALE
MAJOR	C	C E G	C D E G A (MAJOR PENTATONIC)
MAJOR 7	$C^{\Delta 7}$	C E G B	C E F G A B (MAJOR, NO 9TH)
MAJOR 9	C^{9}	C E G B D	C D E F G A B (MAJOR)
SEVENTH	C^{7}	C E G $B^{\flat}$	C D E F G A $B^{\flat}$ (MIXOLYDIAN)
FLAT-9	$C^{7\flat 9}$	C E G $B^{\flat}$ $D^{\flat}$	C $C^{\sharp}$ E F G A $B^{\flat}$
SHARP-9	$C^{7\sharp 9}$	C E G $B^{\flat}$ $E^{\flat}$	C $C^{\sharp}$ $D^{\sharp}$ E F G A $B^{\flat}$
AUGMENTED	C^{+}	C E $G^{\sharp}$	C D E F $G^{\sharp}$ A B
MINOR	C^{-}	C $E^{\flat}$ G	C $E^{\flat}$ F G $B^{\flat}$ (MINOR PENTATONIC)
MINOR FLAT-6	$C^{-\flat 6}$	C $E^{\flat}$ G $A^{\flat}$	C D $E^{\flat}$ F G $A^{\flat}$ $B^{\flat}$ (AEOLIAN)
MINOR 7	C^{-7}	C $E^{\flat}$ G $B^{\flat}$	C D $E^{\flat}$ F G A $B^{\flat}$ (DORIAN)
MINOR 9	C^{-9}	C $E^{\flat}$ G $B^{\flat}$ D	C D $E^{\flat}$ F G A $B^{\flat}$
MINOR MAJOR-7	$C^{-\Delta 7}$	C $E^{\flat}$ G B	C D $E^{\flat}$ F G A B (MELODIC MINOR)
DIMINISHED	C°	C $E^{\flat}$ $F^{\sharp}$	C D $E^{\flat}$ F $F^{\sharp}$ $A^{\flat}$ A B (WHOLE-HALF DIMINISHED)
DIMINISHED 7TH	C°	C $E^{\flat}$ $F^{\sharp}$ A	C D $E^{\flat}$ F $F^{\sharp}$ $A^{\flat}$ A B (WHOLE-HALF DIMINISHED)

representation extremely difficult.

The second attempt was to load MIDI files and perform largely the same transformation as the spectrogram representation, but without the excess noise. This resulted in inputs that were eight times smaller at $(128, t)$, and made learning feasible, however, the issue with this format was that much of the information in the data pertained to the octave of a particular note. Additionally, the time-step information is largely ignored, since the problem is time-independent. Learning is still possible with this data, but it requires large models to obtain significant results. Since chords are the same no matter which octave their notes fall, normalizing the notes to one octave creates a much smaller input space. Additionally, aggregating all the time steps into a single time step while keeping the ratio of note occurrences and note durations removes the time dependence and allows for much more efficient learning.

3. Models Used

Three different models were tested throughout the training process: a simple Multilayer Perceptron Network (MLP), a Convolutional Neural Network (CNN), a Recurrent Neural Network (RNN). The reasoning behind using the CNN was rooted in the idea that CNNs are known to perform well with image processing, as they are able to extract two-dimensional features from the input that would otherwise be lost in a one-dimensional representation. Utilizing a spectrogram image, or midi representation, of the sound input with the CNN would allow for time-frequency features to be extracted from the input. While CNNs are good for image processing, RNNs, on the other hand, should work better for time-series data. Since music can be thought of as a time series of note values, the RNN should theoretically perform well with this problem.

3.1. CNN Architecture

The CNN was used for the spectrogram and MIDI inputs, and therefore the input sizes were (n, t), where n is the number of notes/frequencies considered, and t is the number of time steps. Architecturally, the CNN consists of three stacked convolution layers with a kernel that only spans the time axis. This is because notes that are close in pitch do not correlate to chords that are similar and therefore cannot be convolved together. An example of this would be the major and minor chords for C: (C E G) and (C $E^{\flat}$ G). The E/$E^{\flat}$ pitches are the sole feature deciding whether the chord is major or minor and, if averaged, would make the model perform worse. After each layer, a ReLU activation is applied, followed by batch normalization and channel dropout with a probability of 20% to reduce overfitting. The output to the convolution layers is flattened and passed into feed-forward layers before outputting a probability for each chord quality.

3.2. RNN Architecture

The RNN was also used for spectrogram and MIDI inputs, but instead of convolving the time axis, each time slice was passed through stacked, fully connected recurrent layers. Inspiration for model architecture was drawn from (Silva and Turchet, 2024), however, the number of layers and the sizes were changed to better fit this problem. In the end, a stacked configuration of an RNN, GRU, and LSTM was used where each had 4 internal layers of 64 nodes each.

3.3. MLP Architecture

Once the input size was reduced to $(12, 2)$, there was no need to use CNNs or RNNs. This significantly simplified the model requirement as the feature space was small enough

to be trained without dimensionality reduction, and thus a simple MLP network was used. The MLP model first flattens the input into a $(24, 1)$ before passing it through 5 feed-forward layers, all except the last with 64 nodes. As with the CNN, a ReLU activation with batch normalization and dropout is applied after each layer.

4. Results

Of the three models, the RNN was unable to learn significantly and therefore performed the worst. Due to the complexity of the RNN model, it is unclear whether the issue is theoretical or implementation related, but because each time slice is significantly smaller than the duration of a single note, most time steps have no changes. Furthermore, the target labels themselves are not time-correlated, as the chord is assumed to span the entire duration of the track. With both of these considerations in mind, it makes sense that a time-heavy model such as an RNN would not learn significantly over this dataset.

The two remaining models are somewhat more difficult to compare, as each has its own purpose. The CNN is useful over the full MIDI-time representation, as the convolution layers allow it to learn over the much larger input size. The large kernels over the time axis do not transfer to the $(12, 2)$ occurrence-duration representation, which is where the MLP model was utilized.

Since the final model was the MLP with the occurrence-duration data representation, the rest of the results will be focused around just these.

4.1. Training Results

When trained for 50 epochs, the model achieves full learning of the training dataset, with a 0% misclassification rate over the validation dataset. When applied over the test dataset, however, the misclassification rate jumps to 42%. The full results are detailed in Figures 1 and 2, which both illustrate the model's confusion matrix. Confusion matrices are a useful tool for analyzing the misclassification result, since for every sequence/chord example in the validation and test set, both the predicted chord and chord are detailed. This can show exactly where a the model is making mistakes and that information can provide some insight into how the model is making decisions and where it can be improved.

Looking at the validation confusion matrix found in Figure 1, it is clear that the model has learned what each of these chords are. Since the validation is withheld from the model during training, there is no way for the model to "cheat" the correct answers and therefore had to have learned the information genuinely. This makes the test confusion matrix in Figure 2 more interesting, as the model can be trusted to provide a reasonable prediction. Take, for example, the test sequences that were sampled from the major pentatonic scale, found in the first row of Figure 2. The model primarily predicted both major and minor-7 chords with equal probability. At first glance it appears that the model must be bad at predicting major sequences, but when writing the notes of the major pentatonic scale out it becomes clear what is happening. Table 2 shows that the given alternative predictions for sequences over the major pentatonic scale have a significant overlap with the scale and are thus appear to be valid alternatives to major chords for this scale. In fact, it appears that when predicting E minor-7 over the C major pentatonic scale, the model is inferring additional harmony, as the B that is present in the E minor-7 chord is not present in the major pentatonic scale.

Table 2. Comparisons of Major Test Predictions

NAME	NOTES
C MAJOR PENTATONIC	C D E G A
C MAJOR	C E G
E MINOR 7	E G **B** D
D MINOR 7	D **F** A C

Table 3. Comparisons of Augmented with Minor Major-7

NAME	NOTES
C AUGMENTED SCALE	C D E F $G^\sharp$ A B
C AUGMENTED	C E $G^\sharp$
A MINOR MAJOR-7	A C E $G^\sharp$

Another point of interest would be the samples that are intended to be augmented sequence chords. The model predicts mainly Minor Major-7 chords for augmented sequences, which are generated from a major scale with an augmented 5th. Table 3 shows that the C-Augmented scale contains the notes for an A-Minor Major-7 chord, as the $G^\sharp$ becomes the Major 7 of the A-Minor chord. Since the Minor Major-7 chord depends on the A being present, an interpretation that can be pulled from this distinction is that Minor Major-7 chords can work with augmented melodies as long as the root of the chord is present sufficiently in the song.

The last point of interest is regarding the prediction of Flat-9 chords over diminished sequences. Diminished scales have a unique property that starting the scale from each note will yield another diminished scale. Because of this every chord that is within the diminished scale is in theory replicated 4 times, as shown in Table 4. Additionally, the Flat-9 chord covers 5 notes of the diminished scale, instead of the 3 notes that the diminished triad covers, or the 4 notes the diminished-7 chord covers. The fact that the model seems to prefer the Flat-9 chord seems to indicate that when given the

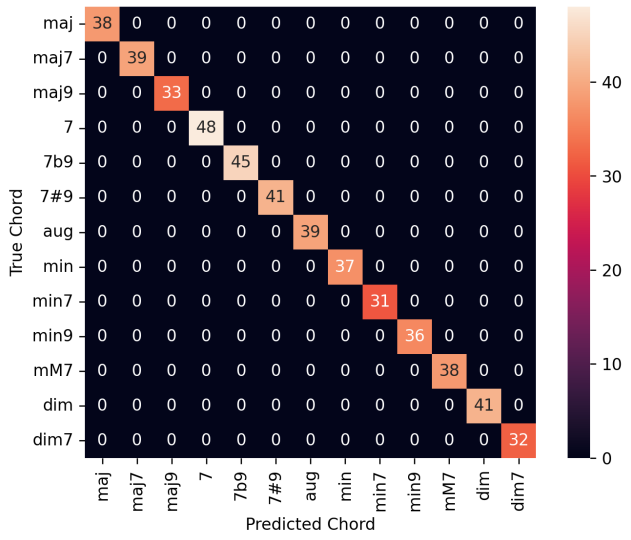

Figure 1. Validation Confusion Matrix

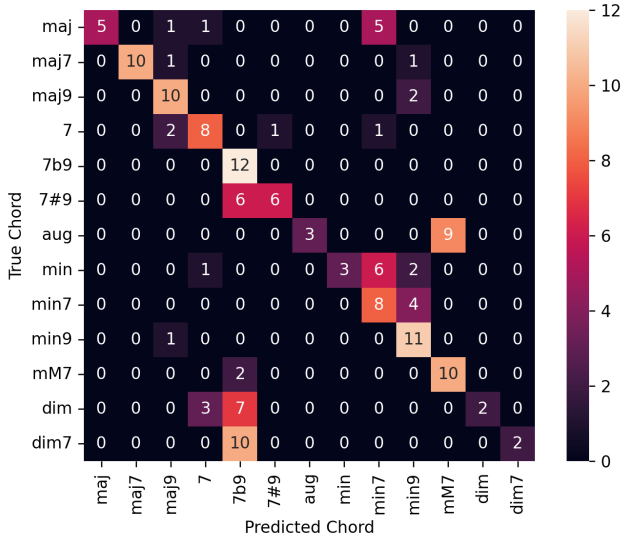

Figure 2. Test Confusion Matrix

Table 4. Comparisons of Diminished with Flat-9 Chords

NAME	NOTES
C DIMINISHED SCALE	C D $E^{\flat}$ F $F^{\sharp}$ $A^{\flat}$ A B
C DIMINISHED 7	C $E^{\flat}$ $F^{\sharp}$ A
$D^{7\flat9}$	D $F^{\sharp}$ A C $E^{\flat}$
$F^{7\flat9}$	F A C $E^{\flat}$ $F^{\sharp}$
$A^{\flat7\flat9}$	$A^{\flat}$ C $E^{\flat}$ $F^{\sharp}$ A
$B^{7\flat9}$	B $E^{\flat}$ $F^{\sharp}$ $A^{\flat}$ C

option between competing chords, the model will choose chords that are richer in nature, as they cover more of the notes that are in the melodic passage.

5. Conclusion

Music at its core involves the subjective interpretation of secretly objective structures, and because of that, is the perfect candidate for machine learning tasks. This project demonstrates that it is possible to take melodies and assign chords to the melodies that can be justified harmonically with a reasonable degree of accuracy. It is both able to classify melodies with their true chords, as well as suggest chords that have different harmonic interpretations within the same melodic framework. Additionally, a useful data representation of music was found which results in a low-dimensional embedding of sound waves over time – something that can be utilized for future research.

5.1. Future Work

If the project development were to continue, there are many directions open for research. This model was only trained on synthetic data, and extending the training process to real songs, as well as expanding the possible chord extensions that are predicted would further enhance the usability of the model. Additionally, while some scales and chords are enharmonic equivalents (think C major and A minor), in real music the "correct" scale depends on which notes lie on the "strong" beats of the measure (commonly beats 1 and 3). Because of this, while the time axis was ultimately ignored for this training set, extensions of this model should consider adding time-dependency to the outputs to train further nuance between the different harmonic progressions. A final improvement that could be made to this model would be to generate training data that is missing some of the chord tones. This type of training would push the model to infer harmonic structure outside of what is given to it, and the suggestions that it would give could be more musically surprising while still being ultimately rooted in the melody.

6. Citations and References

References

Sivangi Chatterjee, Srishti Ganguly, Avik Bose, Hrithik Raj Prasad, and Arijit Ghosal. Audio processing using pattern recognition for music genre classification, 2024. URL https://arxiv.org/abs/2410.14990.

Nishal Stanislaus Silva and Luca Turchet. Real-time pattern recognition of symbolic monophonic music. In

Proceedings of the 19th International Audio Mostly Conference: Explorations in Sonic Cultures, AM '24, page 308–317, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400709685. doi: 10.1145/3678299.3678329. URL https://doi.org/10.1145/3678299.3678329.

Drew Edwards, Simon Dixon, and Emmanouil Benetos. Pijama: Piano jazz with automatic midi annotations. *Transactions of the International Society for Music Information Retrieval*, Sep 2023. doi: 10.5334/tismir.162.

Fatemeh Jamshidi, Gary Pike, Amit Das, and Richard Chapman. Machine learning techniques in automatic music transcription: A systematic survey, 2024. URL https://arxiv.org/abs/2406.15249.

Tosiron Adegbija. jazznet: A dataset of fundamental piano patterns for music audio machine learning research, 2023. URL https://github.com/tosiron/jazznet.