Improving Chord Prediction in Jazz Music using Melody Information

Leonhard Driever[*1] leonard.driever@epfl.ch
Mels Loe Jagt[*1] mels.jagt@epfl.ch
Kuan Lon Vu[*1] kuan.vu@epfl.ch
Daniel Harasim[1] daniel.harasim@epfl.ch
Andrew McLeod[1] andrew.mcleod@epfl.ch
Martin Rohrmeier[1] martin.rohrmeier@epfl.ch

[1]École Polytechnique Fédérale de Lausanne (EPFL)
Digital and Cognitive Musicology Lab
[*]Authors 1, 2, and 3 contributed equally to this work.

ABSTRACT

Chord sequence prediction is a well-studied task in music information retrieval that involves predicting the next chord given the sequence of previous chords. It is of practical interest as a part of models for tasks such as chord transcription and automatic accompaniment. Less studied is the interaction between the chord sequence and additional information encoded in the melody. This study investigates whether a long short-term memory network (LSTM) that jointly considers the previous chords and melody notes can achieve higher chord-prediction accuracy than a baseline LSTM that uses only chord progressions as input. The models are trained and evaluated on the Weimar Jazz Database, which comprises transcriptions of melody improvisations over 456 Jazz standards. The results show that the inclusion of melody information significantly improves the chord-prediction accuracy, particularly for rare chords. Qualitative and quantitative investigations of the models' predictions suggest that our model benefits most from melody information in two cases. First, the melody model performs much better in the absence of melody, suggesting that this lack of melody is itself informative. Second, after melody notes that imply rare chords (e.g., notes that occur in diminished or half-diminished chords), the model's performance on those rare chords improves.

1. INTRODUCTION

Harmony plays an important role in a wide variety of styles and genres of music, and the prediction of the next chord in a sequence of chords constitutes a fundamental task in Music Information Retrieval (MIR) [1]. Musicologically, sequential chord prediction is important for the study of the syntax of tonal harmony [2, 3], in particular for advancing the understanding of the structure of chord progressions. From a practical point of view, models for sequential chord prediction are important components in systems performing a variety of tasks, such as chord estimation (from both audio and MIDI) [4–7], harmonic structure detection [8–10], automatic accompaniment [11, 12], and music generation [13,14]. Typically, such systems also have other components such as a segmentation or duration model which predict the locations of chord changes [5, 8]. In this work, however, we use the ground-truth chord segmentation, and instead concentrate purely on the prediction of the next chord in a sequence of chord symbols.

One important axis along which chord sequence prediction models can be categorized is by the chosen model architecture. Skip-gram models, as well as Markovian approaches such as n-gram models, consider a few chords at a time as symbols themselves [15–19]. They typically treat different chords independently rather than using any feature extraction. Multiple viewpoint approaches can account for musical dimensions such as rhythm and pitch separately [19–21]. Non-sequential models, such as generative grammars [22–25], can also be used for sequential chord prediction [26]. While such grammars are strongly rooted in music theory, their practical application typically involves more computational complexity than the aforementioned methods.

Deep learning methods, in particular recurrent neural networks (RNNs), have been the dominant method in recent years due to their flexibility, short development time, and high performance (provided that sufficient training data is avaliable). RNNs are able (in theory) to consider the entire history of chords during prediction [5, 8–10]. They can treat each chord independently (like n-gram models) by using one-hot vectors as input, or can account for multiple musical dimensions simultaneously (like the multiple viewpoint approaches) by using hand-crafted features or multi-hot vectors as input. Additionally, no matter the specific input format and features, chord embeddings can easily be trained (e.g., by Word2Vec [27]) and used in order to learn relationships between similar chords [28, 29].

In this work, we explicitly investigate whether adding information about the past melody (i.e., the melody during previous chords) improves prediction accuracy for the subsequent chords. It has been shown that external, non-chordal information can improve harmonic structure prediction performance, for instance rhythmic and metrical

Root note	∈ ({A-G} × {♭, ♯, Ø})
Triadic form	∈ {maj, min, dim, sus4, aug}
Extension (+ alteration)	∈ {7♭/♯, 9♭/♯, 11♭/♯, 13♭/♯}
Bass note	∈ { \ + ({A-G} × {♭, ♯, Ø})}

Figure 1: WJazzD's basic chord notation. Every chord consists of at least a root note and a triadic form. Extensions (with optional alteration) and bass note are optional.

information [30]. While it is essential to use *concurrent* melodic information (encoded in the form of chroma vectors [5] or MIDI notes [8]) for automatic chord estimation and harmonization, in this work only the melody prior to the chord to be predicted is taken into account.

It is important to emphasize at this point that we do not intend to create a state-of-the-art system for chord prediction, and therefore a comparison with such methods is outside of the scope of this work. Rather, we present here a controlled experiment investigating the effect of a single feature (past melody information) on chord prediction performance. As a consequence, our findings can inform the design of chord prediction systems in real-time scenarios such as live transcription or automatic accompaniment. However, we make here two simplifying assumptions that are not fulfilled for automatic accompaniment: our model takes as input the ground truth chord segmentation boundaries, and the model gets as input the correct chords from the previous steps. These assumptions are essential to eliminating noise from our experiments and enable direct conclusions about the effect of melody on chord sequence prediction.

A motivating idea for this study is that the melody might contain information about the tonal context of a section of music, which leads to more accurate predictions. Consider the scenario of predicting the chord following a G7. If the melody played over the G7 contains the notes E♭ and A♭, then a C minor chord should be more probable than a C major chord, while the notes E and A would more strongly imply a C major chord. The G7 chord on its own would not disambiguate between these two cases. All code for the models discussed in this work are available online. [1]

2. DATA

2.1 Dataset

In this study, we use the *Weimar Jazz Database* (WJazzD) [31], containing 456 unique jazz solo improvisations. Each solo improvisation contains a transcribed improvised solo melody and the chord accompaniment. Of the tables provided in the WJazzD, only *melody* and *beats* are selected and utilized for further analysis, providing information on the pitches in the melody and the accompanying chord progression respectively for all solos.

In the WJazzD, a chord is defined by a root note (pitch class), a triadic form, extension (and alteration), and the bass note (see Figure 1). The chord extension is any ascending combination of 7, 9, 11 and 13, each accompanied by an optional alteration (flat or sharp). The bass note indicates the lowest note in the chord and is assumed to be the same as the root note if not included. We treat all enharmonically equivalent pitch classes as identical (i.e., Ab and G# are the same).

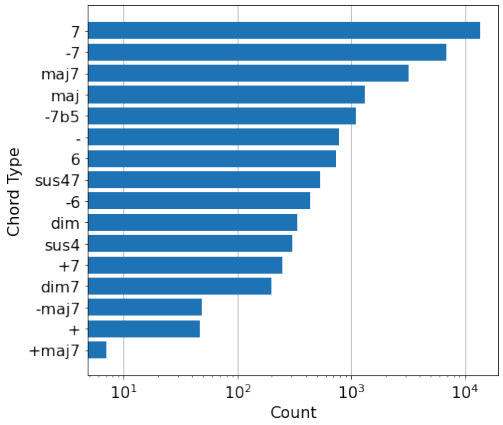


Figure 2: Number of chords per type in the full WJazzD after type reduction.

Each jazz solo is referenced by a unique ***melid*** $\in [1, 456]$. Together with a ***bar*** $\in [-1, N_{\textbf{\textit{melid}}}]$ and ***beat*** $\in [1, 4]$, a unique coordinate to identify a distinct beat in a particular solo is formed. Here $N_{\textbf{\textit{melid}}}$ indicates the max number for ***bar*** specific to a solo. Chords in the WJazzD only change on beats. Thus, utilizing this coordinate system, for each beat in a solo, its corresponding chord can be selected. Each beat also has a duration, measured in seconds. The WJazzD is not quantized, so consecutive beats typically have slightly differing durations.

Notes from the solo improvisations are also referenced by this system, which we use to assign each note to a particular chord, as well as by more fine-grained sub-beat and tatum coordinates, which we use only for ordering the notes. Each note has a MIDI pitch in the range $0 - 127$ and a duration (unquantized), measured in seconds.

2.2 Reduction of Chords

There are 418 unique chord symbols in the dataset, excluding 'NC' which stands for no chord, and which we handle by continuing the previous chord. Some of these chords are very rare in the dataset. To reduce this sparsity, we alter the chord notations by removing chord extensions beyond the 7th (e.g., Ab+79b → Ab+7; D#7913 → D#7). This reduction maps rare chords to simplified and more frequent representations while preserving their core. After this reduction, 297 unique chords remain. By further removing the bass notes for each chord (e.g., C7/A → C7; A+/C# → A+), the number of unique chord types (triadic form plus remaining extensions) is reduced to 16, and the number of unique chords (including their root) is 183. Figure 2 shows the resulting distribution of chord types, which is still highly skewed, but to a much lesser degree than the

[1] https://github.com/ldriever/ML_Jazz

original dataset. Following this reduction, 4% of all the chords are identical to the previous chord.

3. MODEL

3.1 Input Representation

In this work, we compare two different model input representations. First, a baseline model, for which the input represents only a sequence of chords, and each input vector represents a single chord. The second model investigates the influence of including information about the improvised solo melody by instead encoding each chord as a *sequence* of input vectors, each representing the chord as well as a single note played during that chord (if there are any).

3.1.1 Baseline

For the baseline model, each tune is represented as a sequence of input vectors, each representing a chord without any information about the improvised solo melody.

Each chord is encoded into a multi-hot vector of length 24. The first 12 entries indicate the notes in a chord. We use a multi-hot chroma vector of length 12 with the root of the chord at 0. Thus, each chord of the same type (e.g., maj7) will be identical in their first 12 elements. The second 12 elements indicate the root note of the chord represented by a one-hot pitch class vector with C=0. By encoding chords in this way, we embed them in a space where some information can be shared across related chord types irrespective of their root (e.g., major chords and augmented chords share a major 3rd above the root) as well as across unrelated chord types with the same root (e.g., C7 and Cdim7 share a root note).

3.1.2 Encoding Melody Information

For the melody model, each input vector represents a single note and its associated chord. If multiple notes occur during a single chord, they are encoded in separate vectors with the same chord representation. Here, each input vector is composed of three concatenated parts, of length 45 in total: the chord (identical to that described for the baseline), the pitch, and the duration.

The pitch of a note encoded in a multi-hot vector of length 20. The first 12 elements are a one-hot vector indicating the enharmonic pitch class of the note, where C=0. The last 8 elements encode the octave of the note from 0–7, where we define middle C to be in octave 3. To complete the note's encoding, the last element indicates the duration of the note, measured as a proportion of the corresponding beat. Since notes in the WJazzD are not quantized, this duration is typically not an even multiple or divisor of the beat, but rather include some noise, indicating a human bias in the length of the note played. In the absence of any notes during a chord, the pitch and duration parts of the input vector are all set to 0.

Since we consider each melody as a sequence of notes (represented by their pitches and durations), but the task is to predict the next chord, the melody model's prediction is only measured after the last note of a given chord. Structurally, this leads to one important difference between the baseline and melody models: The melody model may receive each chord as input multiple times, whereas the baseline model only sees each chord once. To control for this difference, we considered duplicating the baseline's input similarly. However, that would in fact still encode melody information into the baseline model—specifically, the number of notes that occur during each chord, which, as we show in Section 4, is a key piece of information that the melody model relies on for its increased performance.

3.2 Model Architecture

Both baseline and melody models have an essentially identical structure based on a recurrent neural network (RNN). Unlike dense or convolutional neural networks, RNNs are able to retain sequential information and can thus be applied to data, such as music, of variable-length sequences where the order of data points is of key importance. The selected type of RNN is a long short-term memory (LSTM) network [32]. An LSTM makes use of two hidden cell states and an arrangement of three types of "gates" to allow it to retain information over longer sequences of input data. This allows LSTM networks to overcome the "vanishing gradient" problem faced by traditional RNNs [33]. We use uni-directional LSTMs because a bi-directional LSTM would require using information from the entire known chord series, which is not compatible with the aim of chord sequence prediction.

The model architecture is depicted in Figure 4, where "v: l" indicates that the data at that point is a vector of length l. It consists of a single dense embedding layer, one or more LSTM layers (set via hyperparameter tuning; see subsection 4.1), and a single dense output layer. The embedding layer finds a suitable representation of the input data and converts each data point to a representation of size "embedding size" (set by hyperparameter tuning). The LSTM layers handle the sequential information of the data and output data points of size "LSTM hidden size" (set by hyperparameter tuning). The final linear dense neural network layer translates the data into the desired representation: a vector of length 183 where the location of the maximum value indicates the predicted chord. Remember that in the case of the melody model, only the output resulting from the last input note of each chord is used.

4. RESULTS AND DISCUSSION

4.1 Hyperparameter Tuning

For our experiments, we performed 10-fold leave-one-out cross-validation so that each tune in the WJazzD can be included in a test set. Thus, 10 different versions of each model were trained, each with one fold for testing, one for validation, and the remaining 8 for training. We tuned the models' hyperparameters on a random 0.8, 0.1, 0.1 split (not aligned with any of the folds), and used the values which maximized performance on that validation set throughout testing.

To train each model, we use the Adam optimizer [34], with cross entropy as a loss function, and weight decay to

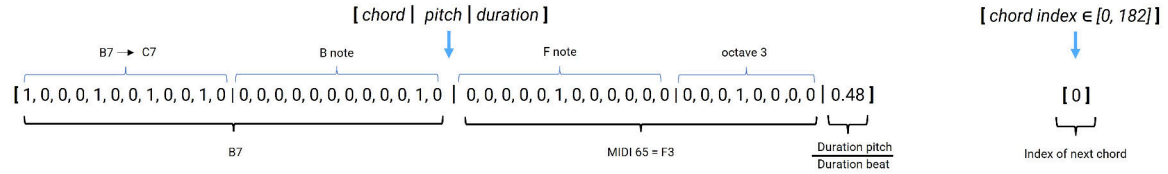


Figure 3: Schematic of input vector (left) and its target (right).

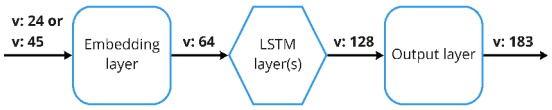


Figure 4: Architecture of the used machine learning model.

implement l_2 regularization. To reduce the amount of overfitting, we apply early stopping to end training when the validation loss has not decreased over the last 30 epochs. To set the batch size and learning rate, we used the automated tuning algorithms of the Pytorch-Lightning module (based in part on [35]) which resulted in a batch size of 64 and a learning rate of 0.014 for each model. Additionally, we optimized the amount of weight decay in the range 10^{-5} to 0.1 for each model, resulting in 0.001 for the baseline and 0.00008 for the melody model.

Finally, we also optimized three structural hyperparameters using a grid search: embedding size $\in \{32, 64, 128\}$, LSTM hidden size $\in \{32, 64, 128, 256\}$, and the number of LSTM layers $\in \{1, 2, 3\}$. The best performing values were 64 and 128 for both models for the embedding size and the LSTM hidden size respectively, and 1 and 2 LSTM layers for the baseline and the melody model respectively. That the melody model is larger was to be expected: It must retain information over longer sequences of input data.

For both models, the training and validation losses remained rather far apart from one another, even at the best identified value of the weight decay. However, this was deemed unavoidable as a sensitivity analysis was performed and showed that any further increases and decreases of the weight decays lead to deteriorated performance in terms of validation accuracy.

4.2 Results

Table 1 reports the average prediction accuracies of the baseline model and the melody model across the entire WJazzD. Note that for the accuracy averaged across each chord, the contributions of different tunes depends on their length. The results show that the additional melody information has lead to an increase in prediction accuracy, from 44.95% to 47.37% when averaged across each chord. For both models, there is a wide spread between maximum and minimum performance when averaged per tune, ranging from $0\% - 100\%$ for the melody model and $0\% - 98.6\%$ for the baseline. Nonetheless, the melody model's increase in performance is consistent across all 10 folds.

To quantify the evidence for the hypothesis that melody information increases prediction accuracy averaged per tune,

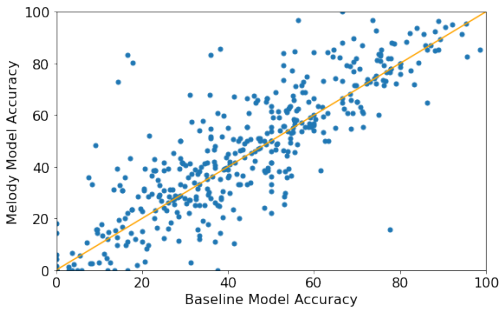


Figure 5: Comparison of prediction accuracy per tune for the baseline and the melody model. The diagonal represents unchanged accuracy.

we performed a Bayesian linear regression that models a piece's accuracy on the basis of the model ID, the tune ID, and the model training ID for each fold (because the chord predictions inside one fold are not independent). We use weakly informative prior distributions over the regression coefficients that weakly favor the alternative hypothesis that melody information does not increase the prediction accuracy. This setup quantifies the uncertainty in how much of the accuracy difference is actually related to the different model architectures by controlling for tune variety and model training (see for example [36, 37] for more information about this statistical testing approach). The regression was implemented via Markov-chain Monte-Carlo sampling using the python package `bambi` [38], which is based on the probabilistic programming language `pymc3` [39]. The results show that the probability that the accuracy differences can be traced back to the model architectures is about 99.52%. In other words, it is over 200 times more likely that melody information increases chord-prediction accuracy than not. This can be considered very strong evidence [40].

Each model's performance per tune is visualized in Figure 5, where points above the diagonal indicate that the melody model improves prediction accuracy compared to the baseline, and points below the diagonal indicate that prediction accuracy decreases for the melody model. Overall, melody information improves prediction accuracy for 49.66% of the tunes in the WJazzD (average accuracy increase 11.94%), had no effect for 12.41% of the tunes, and worsened performance for 37.93% of them (average accuracy decrease 9.48%).

For further investigation, we quantified each model's performance on target chords where melody notes are present

Model	Baseline	With Melody
Average across Chords	44.95	47.37
Standard Error	0.29	0.29
Average across Tunes	43.55	45.88
Standard Error	1.09	1.20

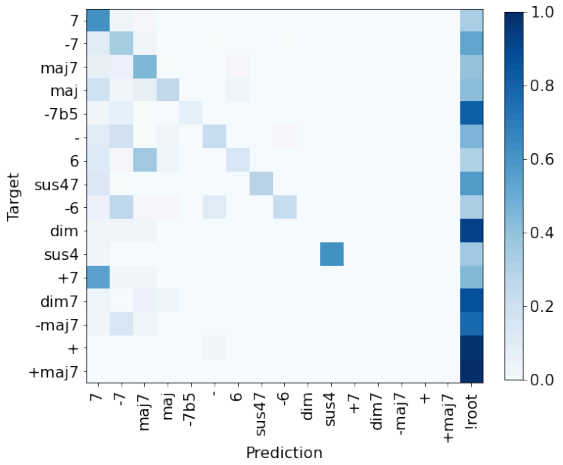
Table 1: Average prediction accuracies in percent for both models across the entire WJazzD.

during the previous chord and those where they are not. Interestingly, the melody model outperforms the baseline by 2.26% when a melody is present, but by 3.80% when it is not. This is a counter-intuitive result: the melody model improves most when there is no melody. However, it appears that the absence of a melody is also a strong signal that the baseline model misses.

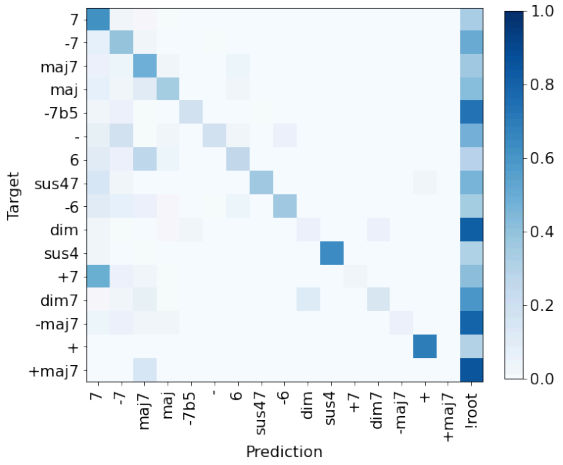
A confusion matrix for the baseline model's predictions is shown in Figure 6a, and the melody model's confusion matrix is shown in Figure 6b. Each of these matrices is normalized by row, and "!root" refers to predictions which have an incorrect root and are therefore not included in any other column (the other columns only capture predictions where the root is correct). Chord types are sorted by frequency, and the figures show that both models perform very well on the most common chord type (7), and worse on the more rare chords, which is not unexpected. However, the melody model sees an improvement over the baseline in the more rare chords, especially from maj to dim. This is visualized in Figure 6c, which plots the difference between the two confusion matrices (melody minus baseline model) using a diverging color scale. Here, the improvement of the melody model in correctly predicting chords is shown by red entries along the main diagonal. The blue entries in the "!root" column (the rightmost column in Figure 6) clearly show that the melody model also reduces the proportion of errors due to an incorrect root. In fact, the only chord type for which the baseline model outperforms the melody model is minor triads, which the melody model sometimes misclassifies as rarer 6th and minor 6th chords (although this is not a common mistake, as can be seen in Figure 6b).

To investigate what specific aspects of the melody tend to correlate with the increased performance of the melody model, Figure 7 shows how the presence of each pitch during the chord preceding each target chord influences each model's prediction accuracy. The pitches are denoted by their interval above the root of the target chord. For example, a C7 for which the melody included an A and a B during the previous chord would count towards the "Pitch present" values for M6 and M7 and the "Pitch absent" values for all of the other intervals. The values for the intervals m3, d5, and m6 indicate that the melody model outperforms the baseline to a much greater degree following seeing those pitches than in their absence.

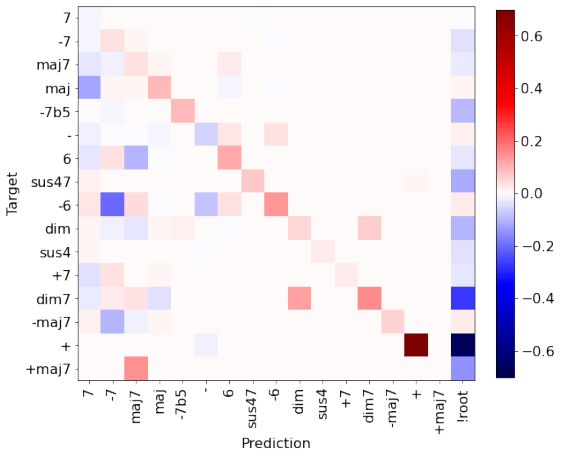
In fact, for these intervals (the three largest increases), the results have a clear music theoretical explanation: after a d5, the proportion of -7b5, dim, and dim7 chords (all of which contain the pitch a d5 above their root) increases to 9.12% versus 5.50% overall. The melody model already



(a)



(b)



(c)

Figure 6: Confusion matrix for the baseline model (a) and the melody model (b), normalized by row. "!root" indicates a prediction with the incorrect root note, which is not included in any other column. The chords are sorted by their frequency in the WJazzD. The difference (b) minus (a), i.e., the melody model's improvement over the baseline, is visualized in (c).

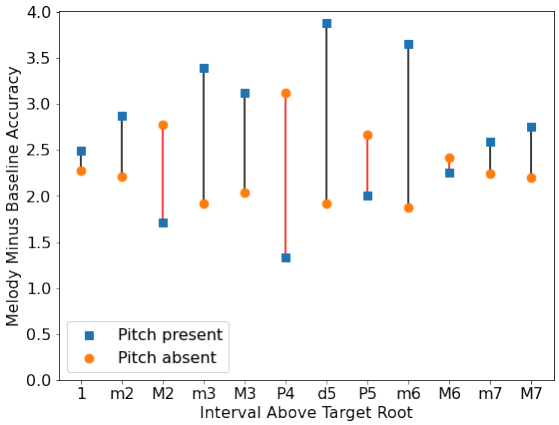


Figure 7: The melody model's improvement over the baseline depending on what pitches (measured by their interval above the target chord's root) co-occur with the chord preceding the target chord (i.e., those which are in the melody model's input).

does better than the baseline on these chords (see Figure 6c), but after a d5, the melody model outperforms the baseline *even more*, by 11.46% compared to 9.32% overall. A similar result is found after m3 and m6 intervals (strong indications of a minor key): the frequency of - and -7 chords is greater after these intervals, and the melody model's increase in performance rises from 3.48% overall to 4.47% in their presence. After a P4, the melody model outperforms the baseline by less than usual on all of the chords rarer than - (and about equals its usual improvement on the more common chords), perhaps because the P4 is both quite common, and not a strong indicator of any of the rare chords music theoretically. The melody model still outperforms the baseline by about 1.3, even in this case.

4.3 Examples

Here, we present four examples of the melody's effect on chord predictions. First, Kenny Dorham's 1955 improvisation over the tune *Lady Bird* [31], for which melody information improves overall performance significantly. The baseline model achieves an accuracy of 27.28% on this tune, while the melody model correctly predicts 45.45% of the chords. This tune is a clear case in which the lack of melody has improved model performance. Bars 15 and 16 (see Figure 8) contain the turnaround *Abmaj7* → *Dbmaj7* → *Cmaj7*. No melody is present for the prediction of these chords. However, as is shown in Figure 8, the melody model correctly predicts each of these chords, while the baseline model misses all of them. This suggests (as we described earlier quantitatively) that the melody model also has the advantage of being able to encode where no melody is played, and to make use of this additional signal.

One example for which melody information has decreased overall performance is Joshua Redman's 1994 solo over the tune *Sweet Sorrow* [31]. For this tune, the baseline model achieves an accuracy of 28.57% while the melody model achieves only 11.43%. Inspecting the chord progression and sheet music for *Sweet Sorrow*, one finds that for most of the tune, the chord progression constantly repeats the chords *Eb7* → *Ab-6*. The melody, however, changes. The model without melody information settles into a repeating pattern of *Eb7* → *Ab7* and thus predicts half of this main chord pattern correctly. The model with melody information, on the other hand, predicts a more varying chord pattern. Thus, it appears that in this case the combination of a steady chord pattern and varying melody means that including the melody information more noise than useful information.

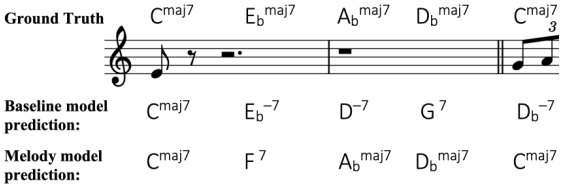


Figure 8: Extract of bars 15–17 from Kenny Dorham's 1955 improvisation over the tune *Lady Bird* [31].

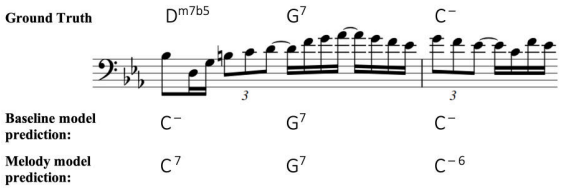


Figure 9: Extract of bars 23–24 from Steve Turre's 1987 improvisation over the tune *Dat Dere* [31].

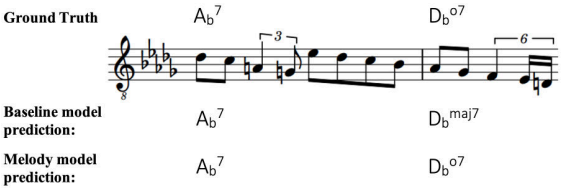


Figure 10: Extract of bars 12–13 from Joe Henderson's 1991 improvisation over the tune *U.M.M.G.* [31]

Looking into specific outputs, Figure 9 shows an excerpt from Steve Turre's improvisation over *Dat Dere*. Here, the melody model predicts C-6 for the final chord, while the baseline correctly predicts C-. As mentioned in regards to Figure 6, this is one example where the melody model overpredicts a more complicated chord, in particular the - → -6 mistake. Furthermore, related to Figure 7, this example shows a case in which a P4 in the melody (the two Fs on beats 3 and 4 of the first bar, which are a fourth above the target C), do not imply any chord in particular.

Finally, Figure 10 presents an excerpt from Joe Henderson's improvisation over *U.M.M.G.*. Here, the melody model correctly predicts the final Dbdim7, while the baseline instead guesses Dbmaj7 (a chord which occurs multiple times previously). Again related to Figure 7, this is one example where a d5 in the melody (the G in the first bar—enharmonic to Abb—which is a diminished 5th above the target Db) appears to have been particularly informative for the model.

5. CONCLUSION

In this work, we presented two simple LSTM models for the task of chord prediction in Jazz, differing only by their input representation and tuned hyperparameters. We have shown that including improvised melody information in the input data leads to more accurate chord predictions. The main increase in performance is on the more rare chords, and this improvement is even more evident when the melody itself contains a note which implies a rare chord. We also showed that, for the melody model, the absence of melody was itself an informative signal.

Future work could investigate whether a similar finding is true for other genres of music. Since Jazz improvisations can be more unrelated to the underlying chord progression than melody from other styles of music (e.g., Western classical or Pop music), it is an open question as to whether a similar improvement can be found for those styles. Including the specific performer as an additional input feature (the WJazzD contains multiple improvisations by each performer) could also test whether the melody model would be able to perform lick detection or improvisation style detection. The information of the performer would help to differentiate melody cues commonly used by each performer, which could lead to more accurate chord prediction. Finally, integrating melody information into a system which must also perform chord segmentation and does not have access to the correct prior chords would be a logical next step, along with updating the model to handle more complicated polyphonic structures that are important to harmonic structure [41].

Acknowledgments

This project has received partial funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under grant agreement no. 760081-PMSB. It was also partially funded through the Swiss National Science Foundation (SNF) within the project "Distant Listening – The Development of Harmony over Three Centuries (1700–2000)" (grant no. 182811). The authors thank Claude Latour for supporting this research through the Latour Chair in Digital Musicology at EPFL.

6. REFERENCES

[1] M. Schedl, E. Gómez Gutiérrez, and J. Urbano, "Music information retrieval: Recent developments and applications," *Foundations and Trends in Information Retrieval*, vol. 8, no. 2-3, pp. 127–261, 2014.

[2] M. Rohrmeier and M. Pearce, "Musical Syntax I: Theoretical Perspectives," in *Springer Handbook of Systematic Musicology*, R. Bader, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 473–486.

[3] M. Pearce and M. Rohrmeier, "Musical Syntax II: Empirical Perspectives," in *Springer Handbook of Systematic Musicology*, R. Bader, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 487–505.

[4] H. Papadopoulos and G. Peeters, "Large-scale study of chord estimation algorithms based on chroma representation and hmm," in *2007 International Workshop on Content-Based Multimedia Indexing*. IEEE, 2007, pp. 53–60.

[5] F. Korzeniowski and G. Widmer, "Improved chord recognition by combining duration and harmonic language models," in *International Society for Music Information Retrieval (ISMIR)*, 2018.

[6] Y. Wu, T. Carsault, E. Nakamura, and K. Yoshii, "Semi-supervised neural chord estimation based on a variational autoencoder with latent chord labels and features," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2956–2966, 2020.

[7] D. Temperley, "A unified probabilistic model for polyphonic music analysis," *Journal of New Music Research*, vol. 38, no. 1, pp. 3–18, mar 2009.

[8] A. McLeod and M. Rohrmeier, "A modular system for the harmonic analysis of musical scores using a large vocabulary," in *International Society for Music Information Retrieval (ISMIR)*, 2021, pp. 435–442.

[9] G. Micchi, K. Kosta, G. Medeot, and P. Chanquion, "A deep learning method for enforcing coherence in automatic chord recognition," in *International Society for Music Information Retrieval (ISMIR)*, 2021, pp. 443–451.

[10] N. Nápoles López, M. Gotham, and I. Fujinaga, "Augmentednet: a roman numeral analysis network with synthetic training examples and additional tonal tasks," in *International Society for Music Information Retrieval (ISMIR)*, 2021, pp. 404–411.

[11] I. Simon, D. Morris, and S. Basu, "Mysong: automatic accompaniment generation for vocal melodies," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2008, pp. 725–734.

[12] T. Carsault, A. McLeod, P. Esling, J. Nika, E. Nakamura, and K. Yoshii, "Multi-step chord sequence prediction based on aggregated multi-scale encoder-decoder networks," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2019, pp. 1–6.

[13] G. Brunner, Y. Wang, R. Wattenhofer, and J. Wiesendanger, "Jambot: Music theory aware chord based generation of polyphonic music with lstms," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2017, pp. 519–526.

[14] D. Conklin, "Chord sequence generation with semiotic patterns," *Journal of Mathematics and Music*, vol. 10, no. 2, pp. 92–106, 2016.

[15] F. C. Moss, M. Neuwirth, D. Harasim, and M. Rohrmeier, "Statistical characteristics of tonal harmony: A corpus study of Beethoven's string quartets," *PLOS ONE*, vol. 14, no. 6, Jun. 2019.

[16] T. De Clercq and D. Temperley, "A corpus analysis of rock harmony," *Popular Music*, pp. 47–70, 2011.

[17] C. Finkensiep, M. Neuwirth, and M. Rohrmeier, "Generalized skipgrams for pattern discovery in polyphonic streams," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018, pp. 547–553.

[18] D. R. Sears, A. Arzt, H. Frostel, R. Sonnleitner, and G. Widmer, "Modeling harmony with skip-grams," in *18th International Society for Music Information Retrieval Conference*, 2017.

[19] M. Rohrmeier and T. Graepel, "Comparing feature-based models of harmony," in *Proceedings of the 9th International Symposium on Computer Music Modelling and Retrieval*, 2012, pp. 357–370.

[20] D. Conklin and I. H. Witten, "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, vol. 24, no. 1, pp. 51–73, 1995.

[21] M. T. Pearce, "Statistical learning and probabilistic prediction in music cognition: mechanisms of stylistic enculturation," *Annals of the New York Academy of Sciences*, vol. 1423, no. 1, pp. 378–395, Jul. 2018.

[22] M. Rohrmeier, "The syntax of jazz harmony: Diatonic tonality, phrase structure, and form," *Music Theory and Analysis*, vol. 7, no. 1, pp. 1–63, 2020.

[23] M. Granroth-Wilding and M. Steedman, "A Robust Parser-Interpreter for Jazz Chord Sequences," *Journal of New Music Research*, vol. 43, no. 4, pp. 355–374, Oct. 2014.

[24] D. Harasim, "The Learnability of the Grammar of Jazz: Bayesian Inference of Hierarchical Structures in Harmony," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, Switzerland, 2020.

[25] D. Harasim, M. Rohrmeier, and T. J. O'Donnell, "A Generalized Parsing Framework for Generative Models of Harmonic Syntax," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018, pp. 152–159.

[26] S. A. Herff, C. Finkensiep, and M. Rohrmeier, "Hierarchical syntactic structure predicts listeners' sequence completion in music," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 43, Vienna, 2021, pp. 903–909.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, vol. 26, 2013, pp. 3111–3119.

[28] E. Anzuoni, S. Ayhan, F. Dutto, A. McLeod, F. C. Moss, and M. Rohrmeier, "A historical analysis of harmonic progressions using chord embeddings," in *Sound and Music Computing Conference (SMC)*, 2021, pp. 284–291.

[29] 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*, 2016.

[30] D. Harasim, T. J. O'Donnell, and M. Rohrmeier, "Harmonic Syntax in Time: Rhythm Improves Grammatical Models of Harmony," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, Delft, 2019.

[31] M. Pfleiderer, K. Frieler, J. Abeßer, W.-G. Zaddach, and B. Burkhart, Eds., *Inside the Jazzomat - New Perspectives for Jazz Research*. Schott Campus, 2017.

[32] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, 1997.

[33] P. Le and W. Zuidema, "Quantifying the vanishing gradient and long distance dependency problem in recursive neural networks and recursive LSTMs," in *Proceedings of the 1st Workshop on Representation Learning for NLP*, 2016.

[34] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[35] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2017, pp. 464–472.

[36] D. J. Barr, R. Levy, C. Scheepers, and H. J. Tily, "Random effects structure for confirmatory hypothesis testing: Keep it maximal," *Journal of Memory and Language*, vol. 68, no. 3, pp. 255–278, 2013.

[37] A. J. Milne and S. A. Herff, "The perceptual relevance of balance, evenness, and entropy in musical rhythms," *Cognition*, vol. 203, p. 104233, 2020.

[38] T. Capretto, C. Piho, R. Kumar, J. Westfall, T. Yarkoni, and O. A. Martin, "Bambi: A simple interface for fitting bayesian linear models in python," 2020.

[39] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, "Probabilistic programming in python using pymc3," *PeerJ Computer Science*, vol. 2, p. e55, 2016.

[40] R. E. Kass and A. E. Raftery, "Bayes factors," *Journal of the american statistical association*, vol. 90, no. 430, pp. 773–795, 1995.

[41] L. Wall, R. Lieck, M. Neuwirth, and M. Rohrmeier, "The Impact of Voice Leading and Harmony on Musical Expectancy," *Scientific Reports*, vol. 10, no. 1, pp. 1–8, 2020.