Improving Chord Prediction in Jazz Music using Melody Information

Leonhard Driever[a], Mels Loe Jagt[b] and Kuan Lon Vu[c]
Mentors: Daniel Harasim, and Andrew McLeod
[a] Computational Science and Engineering, [b] Life Sciences Engineering, [c] Computer Science
Digital and Cognitive Musicology Lab
[a,b,c] École Polytechnique Fédérale de Lausanne, Switzerland

ABSTRACT

Chord prediction in jazz is one of the ways in which machine learning can be used in the field of music. Using the jazz songs included in the *Weimar Jazz Database*, this study investigates whether an LSTM model including melody information in the input data can achieve higher chord prediction accuracy than a similar model using only chord progressions as input. A statistical trend is found suggesting that the inclusion of melody information indeed improves prediction performance, but this improvement is not statistically significant. The way in which prediction accuracy changes for specific tunes is analyzed using two example tunes. Our investigation also shows that for the model with melody information there is substantial improvement on root note prediction.

I. Introduction

One way in which machine learning is applied to music is the prediction of chord progressions. Here, the prediction of the chord at time t is dependent on all the music information played until time $t-1$. Long short-term memory (LSTM) networks have been established to be successful in capturing patterns over a long sequence of data points. Therefore LSTM networks are popular among research in music generation and prediction [1].

Jazz musicians play melodies that accompany chord progressions. Motivated by music theory, it is believed that the prediction performance of sequential chord prediction models can be improved by including this melody information in the input data. The motivation of this project is thus to investigate whether providing an LSTM model with both chord progression and melody information is able to increase the accuracy of the chord prediction. To answer this question, we trained and evaluated two similar models. One uses only the information on the chord progression, whereas the other also includes information on the played melody. The hypothesis is that prediction performance will improve when the additional melody information is considered.

II. Data processing

A. Dataset

In this study the *Weimar Jazz Database* (WJazzD), containing 456 unique jazz solo improvisations, is used [2]. Each solo improvisation contains the improvised 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 (note within an octave), the chord quality, extension (+ alteration) and the bass note. (Figure 1). The chord extension can be any combination of 7, 9, 11 and 13 accompanied by an optional alteration. The only condition is that it must be in ascending order (e.g., 7b911 is possible opposed to 97b11). The bass note indicates the lowest note in the chord and it is generally the root note. In the case the latter is not true, the bass note is indicated by a backward slash as illustrated in Figure 1.

Each jazz solo is referenced by a unique id, $\boldsymbol{melid} \in [1, 456]$. Together with a $\boldsymbol{bar} \in [-1, N_{\boldsymbol{melid}}]$ and $\boldsymbol{beat} \in [1, 4]$, a unique coordinate to identify a distinct beat is formed. Here $N_{\boldsymbol{melid}}$ indicates the max number for ***bar*** specific to a solo. Utilizing this coordinate system, for every beat in a solo its corresponding chord and/or associated pitch(es) being played are selected.

For our experiments, we randomly split the dataset into training, validation, and testing sets with proportion 0.8, 0.1, 0.1 respectively.

B. Reduction of Chords

There are 418 unique chord symbols in the dataset, excluding 'NC' which stands for no chord. Some of these chords are very rarely presented in the dataset. To reduce the number of chords, which helps the learning for the model, we reduced the chord notations by removing chord extensions beyond the 7th (e.g., Ab+79b $\rightarrow$ Ab+7, D#7913 $\rightarrow$ D#7). This maps the chord to its simplified and more frequent representation while preserving its core. As a consequence, now 297 unique chords are left. By further removing the bass notes in every chord (e.g., C7/A $\rightarrow$ C7, A+/C# $\rightarrow$ A+) the number of unique chords is even reduced to 183. This procedure produced a final set of chords in the dataset that is less diverse. To visualize the chord distribution, these remaining 183 chords are mapped to their respective chord quality *major*, *minor*, *diminished*, *suspended 4* or *augmented* in (Figure 3).

III. Model

A. Chord Representation

As input to the model, a unique solo is denoted as a unique sequence of individual input vectors representing the chords (with information about the melody). More specifically, an input vector is composed of three parts: *chord*, *pitch* and *duration* (Figure 3).

Root note	∈ {C-B}
Chord quality	∈ {maj, min, dim, sus4, aug/+}
Extension (+ alteration)	∈ {7b/#, 9b/#, 11b/#, 13b/#}
Bass note	∈ { \ + {C-B}}

Figure 1. Chord notation. Every chord consists of at least a root note and a chord quality. Additional extension(s) (+ alteration) and bass note is optional. Moreover, the combined selection of extensions must be in ascending order without repetition. Alterations can be omitted.

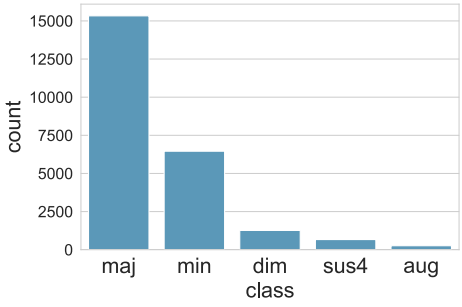

Figure 2. Count plot per chord quality based on the training set (80% of the total data, randomly split).

Chord is encoded into a multi-hot vector of size 24. The first 12 entries indicate the notes in a chord. We use a multi-hot chroma vector of 12 entries with C=0 and B=11 where the first element is the root of the chord. Every chord is also transposed to have C as its root. Thereby, all the notes of the chords in the solos have constant shifts to simplify the pattern for the model to learn. The second 12 elements subsequently indicate the real root note of the chord represented by a one-hot chroma vector. By encoding the *chord* in this way, we embed the different chord types in a space where some notes can be shared (e.g., major chords and augmented chords sharing a major 3rd above the root), and thus bringing them closer to each other. The (MIDI) *pitch* is encoded in a multi-hot vector of size 20 where we define C3 to be the middle C. Here two one-hot vectors are concatenated. The first is of size 12, indicating the pitch class of the note. Similar to the chord representation, the entries range from C-B. The second one-hot vector is of size 8 where the entries refer to the octave ranging from 0-7.

To complete the encoding, the last segment indicates the relative *duration* of the pitch played in the corresponding beat, both measured in seconds. The *duration* of the note is not perfectly quantized as the data is measured in real time, hence indicating a human bias in the length of the note played. In total, the final size of the input vectors is of length 45. Note that if we do not include the melody information, the pitch and duration are omitted. This would result in a total length of 24, that is, the size of the chord encoding only.

Some chords are repeated successively. More precisely, 4% of all the chords (after reduction) being played are the same as the previous chord. Concerning cases where multiple pitches are played within the same beat, we uniquely encode every pitch in a separate vector with each of them associated to the same chord.

B. Model Architecture

As this project deals with data in sequential form, we decided to use 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, where the order of data points is of key importance. The selected type of RNN is a long short-term memory (LSTM) network [3]. These models make use of two hidden cell states and an arrangement of three types of "gates" to allow the model to retain information over longer series of input data. This allows LSTM networks to overcome the "vanishing gradient" problem faced by traditional RNNs [4]. We use one-directional LSTMs because a bi-directional model would require using information from the entire known chord series, which is not compatible with the aim of chord prediction.

The model architecture is depicted in Figure 4, where *v: l* means that each data point in a series is represented as vector of length l. It consists of a dense embedding layer, one or multiple LSTM layers, and a dense output layer. The embedding layer finds a suitable representation of the input data and converts each data point in the sequences to a representation of size "embedding size". The LSTM layers handle the sequential information of the data and output data points of size "LSTM hidden size". 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.

C. Hyperparameter Selection

For the model, the Adam algorithm [5] is used as optimizer, the cross entropy method is used for loss calculation, and weight decay is used to implement l_2 regularization. To reduce the amount of overfitting, early stopping is used to end training when the validation loss has not decreased over the last 30 epochs. For the models we optimized six hyperparameters: the batch size, learning rate, weight decay, embedding size, LSTM hidden size, and the number of LSTM layers. The best performing values for these parameters are summarized in Table I.

The batch size and learning rate were selected by using the automated tuning algorithms of the Pytorch-Lightning module (based in parts on [6]) at an early stage and maintaining the values for all future model configurations. Even though this meant that model training time was not optimized for each run, it allowed for a faster overall process than if the parameters were re-tuned for each run. The lack of re-tuning should not affect the model outcomes.

The other four parameters where tuned using a grid search and tuning was repeated for both the model using chords only, and the model that also includes melody information. This guarantees that both models were evaluated at their maximum achievable performance. The range of considered values for each parameter is also stated in Table I. The best embedding size and LSTM hidden size was the same for both cases, but the model with melody information performed best when a second LSTM layer was included. This was to be expected, as the model with melody information has more data points for each solo and must thus retain information over longer stretches 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 [1]. 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 prediction accuracy.

IV. Results and Discussion

A. Overall Model Performance

The main results of the study are the final prediction accuracies that the two versions of the model achieve on the test set. The

[1]The loss plots can be found on the project GitHub repository: https://github.com/kuanlonvu/ml-project-2-ML-Jazz

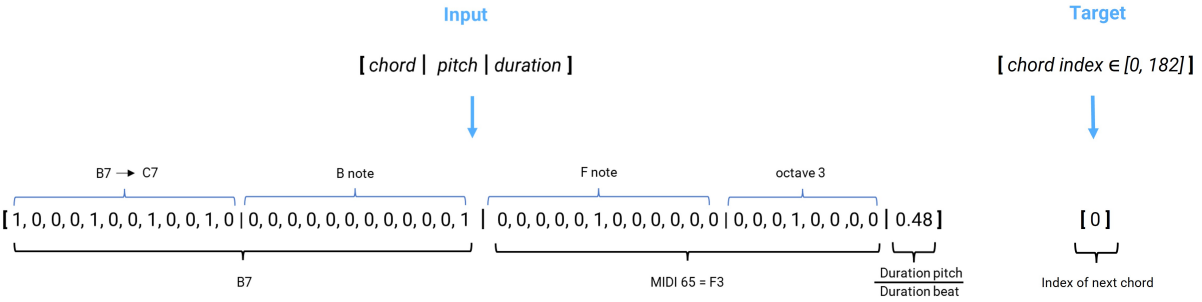

Figure 3. Schematic of input vector and its target.

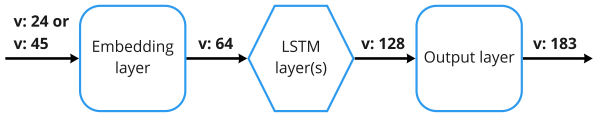

Figure 4. Architecture of the used machine learning model.

Table I
SUMMARY OF MODEL HYPERPARAMETERS.

Hyperparameter	Range	Chords only	With melodies
Batch size	NA	64	
Learning rate	NA	0.014	
Embedding size	32 - 128	64	
LSTM hidden size	32 - 256	128	
LSTM layers	1 - 3	1	2
Weight decay	1e-5 - 0.1	0.001	0.00008

results are summarized in Table II, where SEM refers to the estimated standard error of the mean. The definition of accuracy used here is the overall percentage of correctly predicted chords across the entire set. Thus, the contributions of different tunes to the total average depend on the lengths of the tunes.

Table II
RESULTS FOR THE TWO MODELS.

Quantity	Chords only	With melodies
Total validation accuracy [%]	46.35	55.59
Total test accuracy [%]	43.49	48.54
Highest test accuracy [%]	83.85	93.85
Lowest test accuracy [%]	0.00	0.00
SEM [%]	3.16	3.71

The values in Table II show that the model including melody information in the input data achieves a higher prediction accuracy than the model using only the information on the chord progression. This is visible in the validation accuracy, but also in the more important test accuracy. Here the model with melodies achieves an accuracy that is greater by over five percentage points. However, when assessing the significance of this difference using a permutation test (using 50000 resamples) the determined p value is 0.42. Thus, the improvement in prediction accuracy is not significant. One reason for this may be that the considered test set is too small to provide sufficient evidence. This should be tested in future research under consideration of more data.

Apart from indicating that the model with melody information may be better on average, the data shown in Table II also provides insights into how robust the models are in performing well on new data. For both models, there is a wide spread between maximum and minimum performances. Neither model was able to predict any chords correctly for Steve Coleman's 1989 tune *The Oracle* [2], but both models correctly predicted a fair majority of the chords for Branford Marsalis' 1988 *Housed for Edward* [2].

B. Performance on Individual Solos

We also examined in greater detail how the inclusion of the melodies affects the prediction accuracies for different solos in the test set. This is visualized in Figure 5, where points above the diagonal indicate that the melody-model improves prediction accuracy compared to the chords-only model, and points below the diagonal indicate that prediction accuracy decreases. Points on the diagonal indicate that prediction accuracy is the same for both models. As follows from Figure 5, the model with melodies improved prediction accuracy for 50% of the solos in the test set, had no effect for 20.5% of the solos, and worsened performance for 29.5% of the solos. The largest increase in prediction accuracy is 47.2 percentage points Kenny Dorham's 1955 tune *Lady Bird* [2]. The largest decrease is 28.1 percentage points and corresponds to Joshua Redman's 1994 solo *Sweet Sorrow* [2].

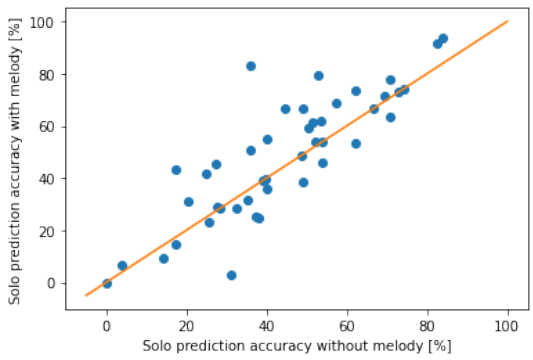

Figure 5. Comparison of prediction accuracies with and without melody information. The diagonal represents unchanged accuracy.

Inspecting the chords and model predictions for the tune *Lady Bird*, which has the largest accuracy increase, one finds that one reason why the melody model is able to improve prediction accuracy is because it correctly predicts the turnaround *Abmaj7* →

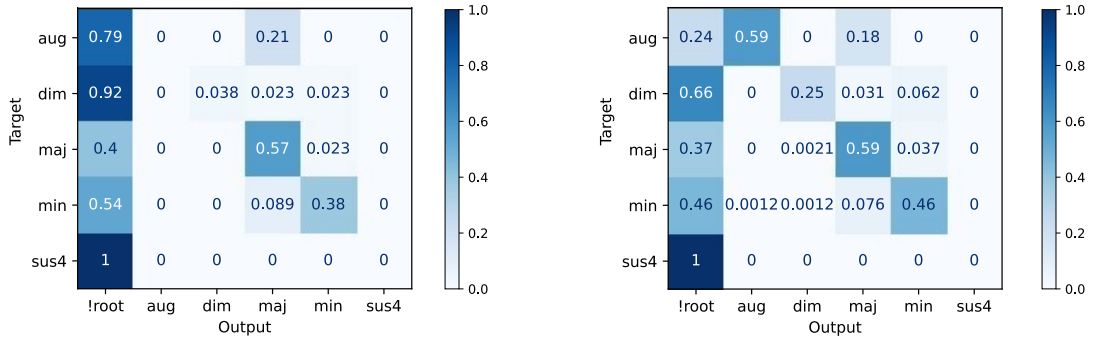

Figure 6. Confusion matrix for the chords only model (left) and melody inclusion model (right) respectively. The values are normalized over the rows. The *!root* column indicates whether the predicted chord has the wrong root note compared to the true chord.

Dbmaj7 → *Cmaj7*. Interestingly, as the extract from *Lady Bird* in Figure 7 shows, this turnaround partly happens while the soloist pauses. This suggests that the melody model also has the advantage of being able to encode where no melody is played and to make use of this information.

Inspecting the chord progression and sheet music for *Sweet Sorrow*, the tune with the largest accuracy decrease, 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 is data noise rather than useful information. However, the possibility must also be considered that this result is simply an artifact of including repeating chords in the input data.

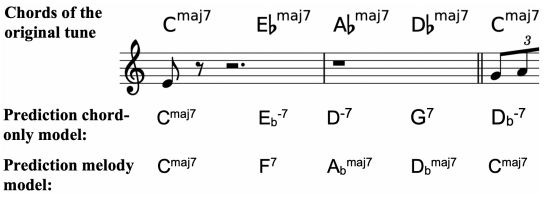

Figure 7. Extract from the sheet music for *Lady Bird*.

C. Performance for Different Chord Types

The performance of both models for different chord types is best summarized using a confusion matrix (Figure 6). We observe a notable improvement when including the melody information, particularly regarding the *!root* column. This group indicates whether the predicted chord has the wrong root note compared to the true chord. As shown, including the melody information substantially improved prediction accuracy on the root note compared to the chord-only model. All the other matrix columns analyze chord quality prediction performance under the condition that the root note is correct. Comparing both models we observe a similar pattern where *maj*, followed by *min*, has the best performance. Interestingly we see that *aug* is not predicted a single time for the chord-only model, and the same applies to *sus4* for both models. This can be explained with regard to the chord type presence as observed in Figure 2. Since the *sus4* chord and *aug* chord are especially underrepresented in the training data, the LSTM models lack sufficient data for this type to properly train.

V. Conclusion

This study has shown that there is a statistical trend that including melody information in the input data provided to a machine learning model allows for more accurate chord prediction. This trend is in line with the hypothesis, but as shown by a permutation test, the difference between the models is not significant. Here, further research using a larger test set is recommended. It was also found that the melody model shows notable improvement on the root note prediction. Another topic for future investigation is which musical reasons can be found in the data for explaining in a generalized way why one model performs better than the other one for specific solos. Such research will then allow a better understanding of the possibilities but also the limitations of using melody information in jazz chord prediction.

VI. Acknowledgements

We thank Dr. Andrew McLeod and Dr. Daniel Harasim for their supervision and guidance in this project.

References

[1] E. Karagül, "Chord Progression and Music Estimation Using LSTM Networks," *Bogazici University*, 1 2019.

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

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

[4] P. Le and W. Zuidema, Eds., *Quantifying the vanishing gradient and long distance dependency problem in recursive neural networks and recursive LSTMs*. The address of the publisher: Proceedings of the 1st Workshop on Representation Learning for NLP, 8 2016.

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

[6] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," 2017.