AN INTERACTIVE COMPUTATIONAL SYSTEM TO ACCOMPANY JAZZ IMPROVISATION

Yiyang Wang
University of Rochester
ywang418@ur.rochester.edu

Joseph Jaeger
University of Rochester
jjaeger5@ur.rochester.edu

ABSTRACT

In this paper, we present an interactive music system where a human soloist performs with a computer agent that provides accompaniment. Currently, the system can generate chords based on both the soloist's melodic material and the computer agent's accompaniment. This allows for the creation of an interactive accompaniment system. While automatic accompaniment algorithms have existed for quite some time, most of these algorithms are designed to provide accompaniment to a predetermined melody and have no interactivity. There are a few music generation models that integrate melody and chord accompaniment, but most of these systems are not designed to function in real-time. In jazz, improvisation is a crucial element. Many "backing tracks" exist on YouTube and other websites where a pre-recorded drum beat, piano chords, and walking bassline are provided. Musicians can then use these backing tracks to practice improvising solos. However, these pre-recorded backing tracks obviously cannot interact with the musician, so the experience is very different from soloing over live accompaniment. Our interactive music system allows a jazz musician to practice improvisation with accompaniment that reacts to their playing and would provide a more realistic and enjoyable experience than a pre-recorded backing track.

1. INTRODUCTION

Improvisation is one of the core elements of jazz. As a soloist improvises a melody, the rhythm section improvises accompaniment. The soloist and rhythm section listen and react to each other, creating a highly interactive musical experience. While improvising, jazz musicians may reference a "lead sheet" that outlines the musical structure and chord progression, but it is common for musicians to make alterations ad lib, especially during solos. Both the soloist and accompaniment must actively listen and react to each other during the performance.

Given the significance of improvisation in jazz, several computational improvisation systems have been developed. Many of these systems aim to create an artificial soloing partner [1- 4]. However, in this project, we aim to create an interactive automatic accompaniment system that reacts to a human soloist and to the previously generated accompaniment. Many existing methods for automatic accompaniment are based on Hidden Markov Models (HMMs) [5, 6], deep learning methods [7], or a combination of these methods [8]. Although Markov models are conceptually simple and can learn from sparse training data, deep neural networks (DNNs) can capture high-order dependencies and may be more suitable for considering long-term temporal structures [9]. In [10], two long short-term memory (LSTM) networks are used to generate polyphonic music. One LSTM predicts the chord progression based on a "chord embedding." The other LSTM uses the predicted chord progression to generate polyphonic music. In [11], Chu et al. create a hierarchical system that comprises three levels. In each level, a Recurrent Neural Network (RNN) generates a component of the song. The bottom level of the system generates a monophonic melody, and the higher levels create the accompanying chords and drum part. However, neither [10] nor [11] are designed to generate accompaniment in real-time and do not interact with a human musician.

In our project, we use RNN to create an interactive music system. The system predicts chords based on the human player's improvised performance and the previous chords generated by the system, and uses the prediction to generate an accompaniment pattern for the human performer. The system is non-discriminative of performance scenarios and can provide accompaniment to a solo whistler and a full ensemble alike. This system can be a great tool for jazz musicians to practice improvisation and may be especially useful to beginners.

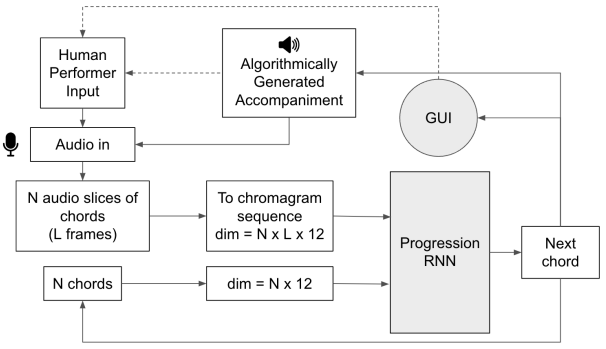

Figure 1. The interactive jazz improvisation accompaniment system. The top half of the figure roughly represents the physical environment (including the interface) while the lower half the virtual data collection and inference process inside the system. Solid-line arrows point in directions of data flow; dotted-line arrows are visual and audio cues provided to the human performer.

2. METHOD

2.1 The Agent

To make the agent interact with the human performer (simplified as *the performer* in the remaining part of the paper) in real time, the interactive improvisation accompaniment agent has an audio input to take in the live performance (which includes the agent's generated output as well) , an audio output for the agent's generated accompaniment, and a graphical user interface to report the status of the system to the performer (Figure 2). User controlled parameters (Settings and Instruments in the GUI) can be set at any time. Once a session starts, the agent generates a rhythmic accompaniment based on the continuous chord predictions. The harmonic content of the accompaniment and the visual cue of the chords will offer an agenda for the performer to interact with, while the groovy rhythm of the accompaniment as well as the visual beat cue shall guide the performer to stay in tempo.

The algorithmically generated accompaniment has three instrument groups that the performer can choose independently. The piano part plays the block chords read from the prediction with a controlled rhythmic scheme of a simple first-order markov chain and stochastically assigned dynamics. The drum part, including kick, snare and two variants of hi-hat, loops over a 4-measure randomly generated pattern. The walking bass line tones are randomly chosen from the chord prediction with a pick-up note offset. Depending on which instrument the performer plays, these accompanying instruments can be freely combined to mimic a more realistic soloing scenario.

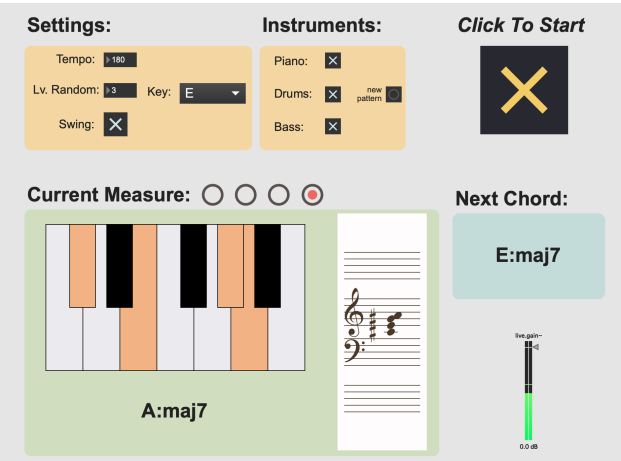

Figure 2. The current graphical user interface of the online improvisation accompaniment system, made in Max8.

Despite some degree of flexibility with certain parameters to be set by the performer, our system imposes quite a few fixed musical parameters. It assumes that a chord stays unchanged for a full measure, and that each measure has four beats. It also assumes one single tonic key throughout a session and accommodates only modulations to closely-related keys that are present in the existing jazz repertoire[1]. Since the system is expected to generate accompaniment in the style of jazz, we consider the above limitations by-and-large conforming with the general stylistic features we aim to match.

One benefit of imposing the said limitations is that we can use the key information from the user to transpose the input and output to conform with our single-keyed model, in which all training data are transposed to C. Another huge benefit is that the scheme simplifies the scheduling and data collection-transportation process greatly. Since the tempo is set, we collect the audio input into a buffer and obtain audio slices by the duration of a measure. The chromagram for each audio slice is computed and stored in sequence, as is the sequence of the output (prediction) chords with the matching indices. Thereby, we are able to utilize the ordered sequential data of the past, in both time and symbolic domain, for the prediction of the next chord. To counteract the latency and offer in-time chord prediction cues for the user, we only take the first 2 beats of the current measure along with the full chromagram data of previous (N-1) measures to predict the next chord, thus trading off a certain degree of accuracy of the prediction for the promptness and capability of the system.

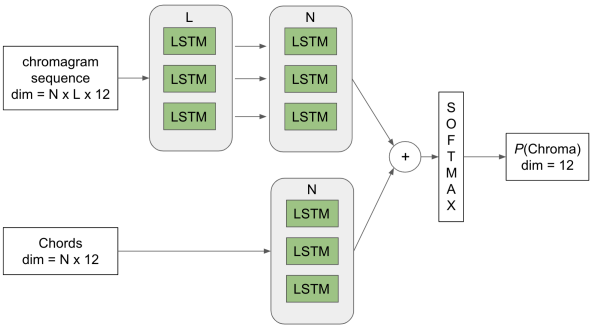

Figure 3. The architecture of the progression RNN.

2.2 Progression RNN

We use recurrent neural networks (RNN) to train a model that predicts the next chord given an audio-extracted chromagram sequence of a jazz recording and the chord labels of the corresponding chord progression. For the inputs, 1) the audio chromagram is sliced such that each chromagram is extracted from the audio portion corresponding to one chord label, and are padded and stacked in order to make a batch that represents the sequence of progression, while 2) the input of the chord sequence is represented as a matrix formed by columns of 12-dimensional multi-hot pitch-class vectors.

For the audio chromagram sequence, the first input of the RNN, each chromagram slice is first fed into a LSTM layer to output the extracted features of each chord-sliced chromagram, resulting in one set of features per duration of chord. Then the sequence of these chord-bounded features are fed into another LSTM layer to output the audio correlated features of the chord sequence.

[1] To be more precise, the system can offer more reliable predictions of modulations to closely-related keys present in the repertoire of early jazz standards, as dictated by the scope of our dataset (Figure 4)

Meanwhile, the sequence of chord labels go through a two-layer LSTM networks without inner dimensional reduction, and the output features of the two concurrent components are concatenated and reduced through a fully connected linear layer to generate a 12-dimensional vector that represents the probabilities for individual pitch classes to be present in the next chord (**Figure 3**).

After we obtain the probabilities for 12 pitch classes, we compute the cross entropy loss between the output and the probability distribution of the binary chroma representation of chords in our dictionary, and choose the chord that has the lowest loss to be our final prediction and report back to the user. During training, we simply compute the cross entropy loss between the prediction output and the target chord.

3. EXPERIMENTS

3.1 Dataset

To train our model, we used the Jazz Audio-Aligned Harmony (JAAH) Dataset[2] [12]. This dataset contains 113 jazz standards and provides structure, key, chord, beat annotations, as well as extracted NNLS chroma features for each song that were used for the chromagram component of the training inputs.

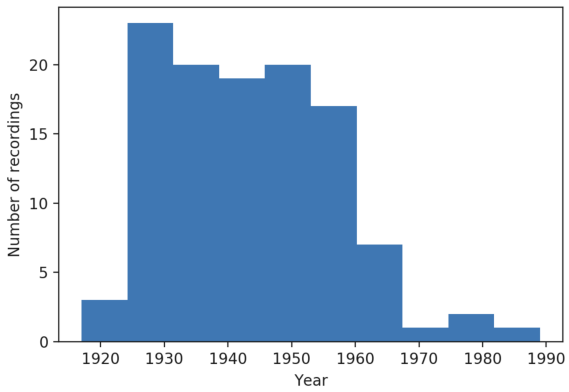

Figure 4. Distribution of recordings by year from JAAH's Github documentation.

From the 113 songs, we exclude six entries with chord annotations failed to parse and one entry that does not have a standard key label, resulting in 106 valid entries in total. 80 songs are used in training and 26 left for validation and testing.

3.2 Training

Based on the assumption that chord progression is invariant with keys, we transpose our training data to the same key for the sake of data augmentation, which is quite easy to achieve given both input and output of our RNN model are chroma-like. For the issue of resolving modal differences, we separate out minor-moded songs from major-mode songs to test for training results based on different grouping strategies, namely 1) train models with major and minor-moded data separately, 2) train one model with modes reduced by parallel relations, and 3) train one model with modes reduced by relative relations. The models trained with only one mode showed lowest loss and thus were used in our live demo. However, it is to be further tested how the models with different training schemes perform.

During training, we use four chords to predict the next chord, and thus have N=4 for the architecture shown in **Figure 3**. The same configuration is used for testing and inner data processing of our online system. The choice is arbitrary and the performance of different configurations could be further tested. For the sake of data augmentation and perturbation, we have four random modes for each iteration of the training, namely 1) feeding both inputs, 2) replacing chromagram sequence input with zeros (and then L1 normalized), 3) replacing chord sequence input with zeros, and 4) skipping the iteration altogether, thus exposing our model to scenarios with partial null inputs and thus increase its fault tolerance.

3.3 Agent Specification

While our model is trained with PyTorch, we use Max/MSP to control the Audio I/O, scheduling, and to provide a GUI for the performer, with data transported back and forth via Open Sound Control protocol. Chroma features are extracted in Max using FluCoMa toolkit with ½ hopsized 2048 frame FFT and L1 normalization. The chroma features are collected and sent to the running Python script via OSC for every beat, with additional time-controlled cues sent for reorganizing chromagram data and making predictions.

4. EVALUATION

It is intrinsically hard to evaluate the performance of a generative model since the fitness and error are hard to judge objectively and statistically. Despite that, several observations are made by the authors regarding the performance of the current version of the accompanying agent.

First, our system usually generates coherent chord progressions that correspond to common jazz idioms. Our system makes frequent use of the ii-V-I progression and circle of fifth motion in general. In fact, if we did not implement a rule to prohibit it, the system would continuously play ii-V-I in the selected key with no human co-improvisor to guide it elsewhere. This is a consequence of the 4-bar window that we used to predict the next chord during training. Since we only considered the previous 4 bars, the model has no knowledge of the fact that jazz chord progressions typically involve many chords and would not continuously cycle through the same three chords. Additionally, the system will occasionally utilize tritone substitutions, but interestingly, we never observed a tritone substitution going to the I chord; tritone substitutions always seemed to tonicize a related key.

While the overall root motions were conventional, the system's choice of chord quality was sometimes

[2] https://mtg.github.io/JAAH/

unorthodox, but largely unoffensive. The system made far more frequent use of modal mixture than was present in the training data, tended to over-use minor-major 7 chords, and may have under-used dominant chords.

Due to the limitations on the scope and timeline of the project, we conducted a feedback survey for a 30-minute live demo following the ECE477 course project presentation where we invited attendees to bring instruments and improvise with our system. Some of the musicians who used our system were experienced improvisers and were impressed with the level of interactivity provided by our system.

Despite the positive feedback from the live demo, perhaps due to the relative sparsity of the training data, the cross entropy loss reported back during the training and validation does not show a significant drop with tweekings of hyper-parameters and more training epochs. It is to be tested in our future work on how to judge the performance of the model statistically.

5. CONCLUSIONS

In its current state, our system generates chords to accompany human musicians in real-time and provides a basic drum beat and walking bassline. In addition, we provide a GUI that displays current and upcoming chord information and allows the user to control several parameters governing the performance. Although it is difficult to objectively assess the performance of our system, those who have used the system agree that the experience is interactive and that the chord generation follows common jazz idioms with only a few minor quirks. To improve the chord generation, we can train separate models for major and minor modes and can seek additional training data, especially for more modern songs. To improve the interactivity and expressiveness of our system, we can integrate models to determine the rhythm of the chords, i.e., the comping pattern, and the dynamics. Nonetheless, our system provides a powerful tool to practice improvisation in its current state.

6. REFERENCES

[1] J. Biles, “GenJam: A Genetic Algorithm for Generating Jazz Solos,” in *International Conference on Mathematics and Computing*, 1994.

[2] B. Thom, “Unsupervised Learning and Interactive Jazz/Blues Improvisation,” in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pp. 652–657, July 2000.

[3] G. Hoffman and G. Weinberg, “Shimon: an interactive improvisational robotic marimba player,” in *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, pp. 3097–3102, 2010.

[4] N. Trieu and R. M. Keller, “JazzGAN: Improvising with generative adversarial networks,” in *6th international workshop on musical metacreation (MUME 2018)*, June 2018.

[5] I. Simon et al., "MySong: automatic accompaniment generation for vocal melodies," in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 725-734. 2008.

[6] N. Ding,. "Research and Design of Automatic Piano Accompaniment System Based on Sound Database," in *International Conference on Forthcoming Networks and Sustainability in the IoT Era*, pp. 121-127, 2022.

[7] K. Kritsis et al., "On the Adaptability of Recurrent Neural Networks for Real-Time Jazz Improvisation Accompaniment." in *Frontiers in artificial intelligence*, vol 3, article 508727, 2021.

[8] T. Hori et al., "Jazz piano trio synthesizing system based on HMM and DNN," in *Proceedings of the 14th sound and music computing conference*, pp. 5-8, 2017.

[9] J. Briot et al. *Deep learning techniques for music generation*, New York, NY: Springer International Publishing, 2019.

[10] G. Brunner et al., "JamBot: Music theory aware chord based generation of polyphonic music with LSTMs," in *IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 519-526, 2017.

[11] K. Choi et al., "Text-based LSTM networks for automatic music composition,” in *1st Conference on Computer Simulation of Musical Creativity*, 2016.

[12] V. Eremenko et al., “Audio-Aligned Jazz Harmony Dataset for Automatic Chord Transcription and Corpus-based Research,” in *International Society for Music Information Retrieval Conference*, 2018.