BachDuet: A Deep Learning System for Human-Machine Counterpoint Improvisation

Christodoulos Benetatos
Department of ECE
University of Rochester
Rochester, NY 14627, USA
c.benetatos@rochester.edu

Joseph VanderStel
Eastman School of Music
University of Rochester
Rochester, NY 14627, USA
j.vanderstel@rochester.edu

Zhiyao Duan
Department of ECE
University of Rochester
Rochester, NY 14627, USA
zhiyao.duan@rochester.edu

ABSTRACT

During the *Baroque period*, improvisation was a key element of music performance and education. Great musicians, such as J.S. Bach, were better known as improvisers than composers. Today, however, there is a lack of improvisation culture in classical music performance and education; classical musicians either are not trained to improvise, or cannot find other people to improvise with. Motivated by this observation, we develop *BachDuet*, a system that enables real-time counterpoint improvisation between a human and a machine. This system uses a recurrent neural network to process the human musician's monophonic performance on a MIDI keyboard and generates the machine's monophonic performance in real time. We develop a GUI to visualize the generated music content and to facilitate this interaction. We conduct user studies with 13 musically trained users and show the feasibility of two-party duet counterpoint improvisation and the effectiveness of BachDuet for this purpose. We also conduct listening tests with 48 participants and show that they cannot tell the difference between duets generated by human-machine improvisation using BachDuet and those generated by human-human improvisation. Objective evaluation is also conducted to assess the degree to which these improvisations adhere to common rules of counterpoint, showing promising results.

Author Keywords

Automatic music generation, Music improvisation, HCI

CCS Concepts

•Human-centered computing → Interaction design; Human computer interaction;

1. INTRODUCTION

While music improvisation today is strongly connected with genres like jazz, it was also a key element of *Baroque* and *Classical* music. Many composers of this period such as J.S. Bach and Beethoven were skillful improvisers, and performers regularly studied improvisation as part of their music education [11]. The crucial role of improvisation is illustrated in the belief that "The whole history of the development of [Western art] music is accompanied by manifestations of the drive to improvise" [6]. Improvisation has also a lot to offer individually to musicians, including, among other benefits, sharpening performers' creativity and critical thinking, increasing their understanding of the musical style they improvise in, and helping them improve their instrument playing skills. Beginning in the 19th century, however, improvisation became a less important part of Western art music; Some of the reasons are outlined in [13].

Motivated by these facts, we developed a system named *BachDuet* that allows a human musician to improvise a duet counterpoint with a computer agent in real time. Our goal was not to create a passive accompaniment system, but a musical agent that would have an equal role with the human performer. Counterpoint fits well to our purpose, since it is, by definition, the art of combining melodies that are *interdependent* in their harmonic implications, while relatively *independent* in their rhythm and contour. Compared to other improvisational techniques such as ornamentation, making variations of a melody, and harmonizing a given voice, the kind of real-time counterpoint improvisation that BachDuet supports is less constrained: no predetermined melody or chord sequence is needed, although it can be incorporated if given. This type of improvisation, however, does need to respect the constraints of Western music theory. In particular, both the human performer and the computer agent are expected to follow common counterpoint rules to achieve a good improvisation result.

A preliminary version of the proposed system was demonstrated in [1], and this paper provides a detailed description of the techniques and experiments. The contributions of this work are threefold: 1) To our best knowledge, this is the first system that allows real-time interaction between a human performer and a computer agent in the classical style. 2) A set of techniques are developed or integrated to build this system, including music representations, key detection and the Graphical User Interface (GUI). 3) Subjective and objective experiments that evaluate the user experience and the quality of the generated music pieces show promising results. Demo videos and code, can be found in our project page [1]

We believe that this effort is an important step toward our goal of "human-computer collaborative music making", and we also hope this system will serve as a practice tool for classical musicians to develop their improvisation skills.

2. RELATED WORK

Automatic Music Generation (AMG) is the task of using algorithms to compose a piece of music. Existing AMG systems include rule-based systems [4], statistical models (e.g. Markov chains [17]), evolutionary algorithms (e.g. genetic algorithms [2]) and learning-based algorithms (e.g. neural

[1] http://www2.ece.rochester.edu/projects/air/projects/BachDuet.html

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). Copyright remains with the author(s).

NIME'20, July 21-25, 2020, Royal Birmingham Conservatoire, Birmingham City University, Birmingham, United Kingdom.

networks [5]). Recent years have witnessed a significant growth in the last category due to the rise of deep learning.

One area of research in AMG is interactive music generation, where a human musician interacts with a computer algorithm (agent) to compose a piece of music [15]. Interaction may happen *offline* which allows iterations in the generation process, or in *real time*, where both the human and the algorithm improvise their music.

In this work we are interested in real-time interactive music generation for counterpoint improvisation. Most work in this category, however, is designed to only support the "call & response" mode, where the user plays for several measures and the machine then tries to respond in an appropriate way [12, 2, 14]. To our knowledge, only one system, named *Voyager* [10], was designed to support "simultaneous play", but the music style is contemporary and the agent only responds to low-level features of the human performance, without any modeling of the high-level musical content.

For real-time interactive music generation of polyphonic music, it is noted that our lab proposed a reinforcement learning algorithm [7] that could support this type of interaction. However, [7] stays at the algorithm level and no real interactions are performed or evaluated. This paper, on the other hand, enables and evaluates such interaction.

3. PROPOSED SYSTEM

The proposed BachDuet system enables a human performer to improvise a duet counterpoint with a computer agent in real time. The concept of this interaction is illustrated in Figure 1. The human performer plays one voice on a MIDI keyboard and the note information is input to the system. The agent employs a deep neural network to improvise the other voice. The music content of both voices is rendered through the GUI, showing both the MIDI piano roll and the music notation. Key modulation detection is performed to estimate note spellings (e.g., C♯, D♭) for typesetting.

The system assumes a 4/4 time signature and operates with an adjustable steady tempo to ensure human-machine synchronization. The extension to other time signatures is straightforward.

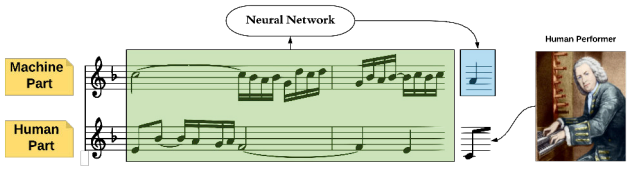

Figure 1: Basic concept of BachDuet.

3.1 Graphical User Interface

The GUI of BachDuet is illustrated in Figure 2. It serves two primary purposes: 1) music content visualization, and 2) system setting control.

Music visualization is important for the user to better understand the interactions. We display both the MIDI piano roll and Western music notation. The piano roll uses different colors to differentiate the two voices. Musical notes that are represented as rectangles originate from the keyboard on the far right and then flow to the left in real time. The entire piano roll panel displays up to about 12 measures of the music. This visualization helps the user see the pitch contours and their interactions in a relatively long temporal scope. The music notation visualization renders musical notes on two staves, treble clef for the upper staff and bass clef for the lower staff. Notes also flow from right to left. However, the rendering of the notes is delayed, as it needs to wait until the note offset signal to determine the duration of the note. Key modulations are estimated and displayed above the music notation to help the system determine the spelling of notes (e.g., C♯ vs. D♭) and accidentals, and to help the user understand the music context. The music score visualization has a temporal scope of about five measures, allowing the user to see more musically relevant information in the recent past.

For system settings, there are several major sections:

- Metronome: The user can specify the tempo and choose different sounds for the downbeat and other beats.
- Human Performance: The user can choose to play on a computer keyboard or an external MIDI keyboard. There are also options for dealing with notes that are shorter than the temporal resolution of our system.
- Neural Network Parameters: There are options to choose different pre-trained models, as well as the amount of randomness of the generated notes (sampling temperature).
- Memory Reset: The user may reset the agent's memory (RNN hidden states) to start a new session.
- Playback Mixer: Through the *Mixer* dialog box, the user can control volume levels of the human's part, the agent's part and the metronome sound.
- File Operations: After a session, the user can scroll the music notation panel to see the entire past interaction, and save the result in a MusicXML file.

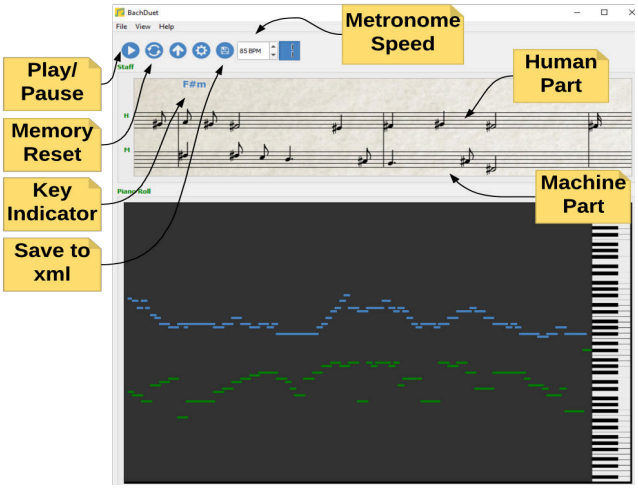

Figure 2: A (truncated) screenshot from BachDuet during operation.

3.2 The Agent Model

We design a recurrent neural network (RNN) to implement the agent that interacts with the human player in real time. We quantize music timing into 16th note steps, and at each timestep, the RNN predicts a token to play. This prediction is performed by sampling the probability of the token conditioned on all previous tokens of both voices.

More specifically, denote the token representations of a duet up to the t-th timestep by

$$\mathbf{d}_{1:t} = \begin{bmatrix} x_{1:t}^{(1)} \\ x_{1:t}^{(2)} \end{bmatrix} = \begin{bmatrix} x_1^{(1)}, & x_2^{(1)}, & \dots & x_t^{(1)} \\ x_1^{(2)}, & x_2^{(2)}, & \dots & x_t^{(2)} \end{bmatrix},$$

where $x_t^{(i)}$ is the token of the i-th part at timestep t. Without loss of generality, assume the first part is played by the human while the second part is by the agent. Our goal is to find a θ-parameterized model G_θ to approximate the

conditional distribution

$$P(x_t^{(2)}|x_{1:t-1}^{(1)}, x_{1:t-1}^{(2)}) \approx G_\theta(x_t^{(2)}|context). \quad (1)$$

Using the hidden state of the RNN as the context summarizer, we can write

$$G_\theta(x_t^{(2)}|context) = G_\theta(x_t^{(2)}|RNN(x_{1:t-1}^{(1)}, x_{1:t-1}^{(2)})). \quad (2)$$

We use 2 layers of 400 Long Short-Term Memory (LSTM) units each for the RNN. Using a variation of the architecture proposed in [8], we augment the RNN by an external stack structure, which serves as an additional memory unit. The stack stores 32 400-d vectors output from the second layer of the RNN. At each timestep, the RNN reads an element from the stack (with attention over all the stack elements), predicts the next token, and then takes an action (push, pop, no action or a combination of these) to update the content of the stack. The final token predictor consists of a fully connected layer and a softmax layer.

3.3 Key Detection

As we mentioned in Section 3.1, one of the features of the GUI is the real-time visualisation of the generated duet in the staff notation format. This requires the system to estimate key modulations for figuring out note spellings and accidentals for the human MIDI keyboard input as well as the agent's Midi_Artic output.

To do this, we integrate another RNN module in our model. This RNN predicts the next key from the music content as well as the previous key predictions. The integration of this key detection module to BachDuet makes the agent model a multi-task learning model. Since key modulations are less frequent, we make the predictions at every beat instead of the 16th note.

For the key unit we used a single layer vanilla LSTM with 300 hidden units. The final key predictor is a fully-connected and a softmax layer with 24 hidden units, each of them corresponding to one of the 24 keys. The input to the key LSTM, is a combination of the previous key, with the hidden state of the note LSTM unit.

3.4 Music Representation

Each $x_t^{(i)}$ token encodes the pitch, articulation and rhythm information of a note. Pitch is encoded with the MIDI number and the chromatic pitch class (CPC), covering a pitch range of [28,94] in MIDI number. The reason for using this encoding instead of a more musically meaningful encoding that incorporates note spellings and accidentals as in [18] is that the human input to our system is through a MIDI keyboard. The latter encoding would require accurate key estimation in real time, and small errors in key estimation may have big impacts on the interaction quality. Rests are given a MIDI number of 0 and a CPC of 12.

For articulation information, we only consider two kinds: onset (1) and hold (0). Notes with a duration longer than a sixteenth note are represented by multiple tokens, where the first is an onset and the rest are holds. We combine the MIDI and articulation encoding to create a new token in the form of [Midi_Artic]. For example, a C4 eighth note is represented by two tokens [*60_1, 60_0*]. This ensures that holds (and onsets) of two different pitches use different tokens. This reduces the token imbalance, as if one common token was used for holds of all the different pitches, then this token would constitute more than 70% of the tokens presented in our training data.

We encode rhythm information following [18]. Specifically, we use three vectors for each timestep: a) a bar vector indicating the beginning of a bar, b) a beat vector indicating the beat division level, and c) an accent vector indicating the metrical accent level relevant to the music style.

In total, we use 135 Midi_Artic tokens (2*67 + 1 for the rest), 13 CPC tokens (12 + 1 for the rest), and 10 rhythm tokens in our encoding dictionary. It is noted that only the Midi_Artic tokens are predicted, while the CPC and rhythm tokens are only fed to the RNN to form the musical context.

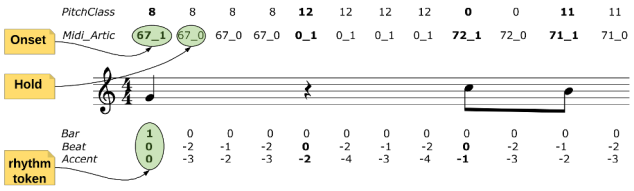

Figure 3: Representation of tokens.

3.5 Embeddings

In order to feed the tokens to the neural network, we have to convert them into numerical values. Taking the Midi_Artic tokens as an example, assigning an integer index to each token according to the pitch order is problematic, as it fails to capture close relations between notes in important note intervals (e.g., octave relations). One-hot encoding (i.e., assigning each token a one-hot vector) has a similar problem; It actually removes all inter-dependencies between notes. We hope to encode our tokens with musically meaningful embedding vectors, where tokens that are close in musical senses are also close in the embedding space. We add embedding layers prior to the RNN layers to automatically learn the embedding vectors from data.

We use three embedding layers, one for each token associated with a timestep (Midi_Artic, CPC, and rhythm). These embedding layers take token indices as input, and use look-up tables to output an embedding vector. These layers can also be viewed as fully connected layers that convert one-hot representation of a token to its embedding vector. Nevertheless, using look-up tables is more efficient. Finally, we concatenate the three embedding vectors to obtain the final embedding vector for each timestep.

The training of these embedding layers is performed jointly with the RNN model in an end-to-end fashion. To confirm that the embedding layers calculate musically meaningful embedding vectors for the RNN to process, after training, we use t-SNE [16], a non-linear dimensionality reduction tool, to visualize the Midi_Artic embedding vectors in a 2D space (Figure 4). A musically trained reader can quickly observe some interesting patterns: There is a clear distinction between the "onset" and "hold" versions of the notes, the rest is isolated in the middle, and notes that have an octave or fifth relation are grouped together.

3.6 Training

3.6.1 Dataset

The proposed agent model needs to be trained on duet counterpoints. We extract the soprano and bass parts of 365 J.S. Bach chorales from the music21 library [3] to form duets. We also transpose each of the duets to all 12 keys (between 6 semitones up and 5 semitones down), totalling 8760 duets.

One may question the validity of the extracted soprano-bass duets in terms of harmonic coherence and completeness, due to the lack of the internal voices. Although this is a valid point, we argue that the issue is less of a concern for developing our agent model. Evidence shows that many soprano melodies in Bach chorales originated from old hymns and folk songs, and when J.S. Bach composed a four- or

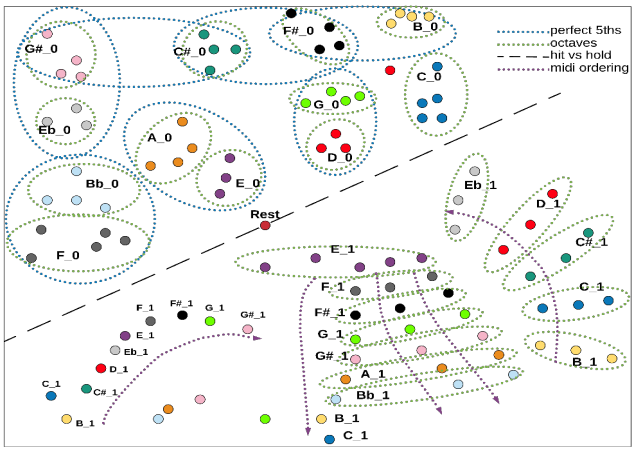

Figure 4: Visualization of the Midi_Artic embedding vectors in 2D using t-SNE.

five-part chorale for them, he first composed the bass voice in a two-part counterpoint with the melody, and then filled the rest of the voices [11]. Nonetheless, it is our future work to include "real" duet counterpoints such as Bach inventions in the training data.

3.6.2 Performance Mistakes Resilience

During real-time operation users may make performance mistakes, or be slightly out of sync with the metronome. The reasons can be either that they are not very capable keyboard players, or that when they improvise they may not be so confident about some notes. Another reason is that human performers tend to play expressively, which introduces micro-deviations from the metronome signal. These mistakes and deviations would introduce "noises" to the token representation during the timing quantization process, and could affect the agent's performance if the training set does not contain such noises. In order to make the neural network resilient to such type of noises, besides the typical regularization methods such as dropout, we also injected four types of common performance mistakes and timing deviations in the training data:

- Time Shift: Randomly shifting a note one 16th note before or after its onset.
- Wrong Duration: Randomly shortening the duration of notes
- Pitch Shift: Randomly shifting a note one semitone or tone higher or lower.
- Deletion: Randomly deleting a note.

In Figure 5 we can see how a melody changes after we inject these mistakes. During training we apply this random noise on every training batch independently, so the neural network is never trained on the same exact data.

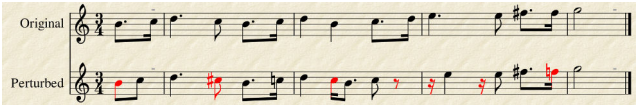

Figure 5: A melody, before and after random mistake injection.

3.6.3 Training parameters

For training, we use the cross-entropy loss and Adam optimizer for both note and key RNN modules. We start with an initial learning rate of 0.001 and reduce it by half every time the validation loss stops improving. We apply dropout of 0.5 on all the layers, excluding the recurrent connections. Finally, we linearly increase the probability of injecting performance mistakes, from 0 to 0.15.

4. EXPERIMENTS

We try to answer the following research questions through a set of experiments: RQ1) Is interactive duet counterpoint improvisation a feasible task for classical musicians? RQ2) Do users find BachDuet a good partner for duet counterpoint improvisation? RQ3) How does the quality of human-machine duet improvisations from BachDuet compare with that of human-human ones?

4.1 Experimental Setup

For the first experiment we recruited 13 musically trained participants, 10 of whom are students at the Eastman School of Music. These participants had passed through a screening process during which they had to answer questions about their music skills. The participants that were qualified were asked to perform two tasks: 1) improvise with our system (HM), and 2) improvise with another participant (HH). For the HH task and for 10 out of 13 participants, the other human partner was a musician who did not belong in this pool, while the remaining 3 interacted with each other. Each task lasted for about 3-10 minutes, and the order of the two tasks was randomized to reduce bias. The users always knew whether they interacted with BachDuet or with another person. Finally, during the HH task, the users were advised not to use any kind of verbal visual or gestural communication, besides the visual feedback from the GUI, to make the comparison between HH and HM more fair.

4.2 User Evaluation

After completing the first experiment, each of the 13 participants were asked to answer a number of questions about the user experience and the musical results:

1. *Did you find the GUI easy/intuitive to use?*
2. *How accurate were the visualizations compared to your actual input from the MIDI keyboard?*
3. *How easy was it to synchronize your playing with the metronome and the machine's output?*
4. *Do you think that the algorithm learnt how to play in the style of chorales?*
5. *How do you describe the direction of information flow between you and the machine (1 from you to the machine while 10 for the opposite)?*
6. *How accurate was the key prediction? (Remember key is different from chord)*
7. *How accurate was the prediction of the note spelling?*
8. *How did you like the musical result of your interaction with BachDuet?*
9. *For the bad musical results, do you feel it was algorithm's fault (1) or yours (10)?*
10. *Do you feel that spending more time playing with BachDuet would improve your improvisation skills?*
11. *Rate how responsive the system was.*
12. *Rate how stimulating/engaging the system was.*
13. *Rate how surprising the system was.*
14. *What is your overall rating of your interaction with the other person?*
15. *What is your overall rating of your interaction with BachDuet?*

The results in Figure 6 indicate that users enjoyed their interaction with BachDuet, yet there is still space for improvement. For the GUI related questions, users were very satisfied with the real-time visualization and the overall GUI. They all preferred the staff notation over the piano

roll. Most of them believed that the algorithm generated music in the style of Bach Chorales. For Q5, score 1 indicates that the information flow is from the participant to the machine, while score 10 indicates the opposite. We believe that a median score around 4.5 from the users was very close to the ideal situation (5.5) where the human and the machine have equal roles in the improvisation. For Q6, even though the key prediction was not always accurate, we can see that the users provided high scores for the spelling of the notes (Q6). This happens because the predicted key, even if wrong, was usually a related one to the actual key (with respect to the circle of fifths), resulting in the spelling of the notes to be correct most of the times.

Users enjoyed the musical result of their interaction with BachDuet to a large extent (Q8), while they felt that both they as well as BachDuet were almost equally responsible for the bad musical results (Q9). For Q10, many participants believed that their counterpoint improvisation skills could be improved using the system, especially if in the future we add a real-time module that indicates mistakes and suggests alternatives. In Q11-Q13, the system was rated very responsive, and their engagement level was high. The surprisement ratings were slightly lower, probably because the agent output did not deviate much from the Chorale's style. Some subjects were more pleased when we increased the sampling temperature (randomness) of our model, but the musical result started deviating from the strict form of Chorales. Finally, BachDuet achieved a high overall rating (Q15) compared to that of human-human improvisations. It is noted that the participants know in advance if they are interacting with the agent or another human; this side information may affect their ratings. Besides the 15 quantitative questions, the participates were also asked to provide open-ended comments. The overall takeaway message from some of these comments is that our algorithm predicted cadences and key modulations better than humans. However, they felt more relaxed when playing with another human, probably because, when interacting with the system, they felt like they had to play continuously without rests. Our observation is that the average harmonic coherence of the duets generated in the HM task was better. However, in the HH task, there was more heavy use of imitative counterpoint, as well as the tendency to repeat some melodic patterns, giving the impression of larger-scale structure.

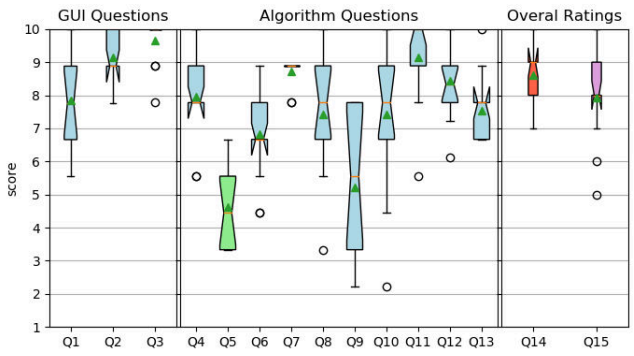

Figure 6: Box plots of scores of all questions received from the 13 participants. For all questions except for Q5 (green color), the scores are the higher the better.

4.3 Subjective Listening Test

For this test, we gathered the generated duets of each of the 13 participants from our experiment, and kept the most harmonically coherent segments from each duet. We managed to gather 40 audio clips of about 1-2 minutes each, split evenly among the HH and HM tasks. Having these, we recruited a new group of participants, and set up a listening Turing test where they had to listen to five randomly chosen duets, rate them, and decide if they correspond to an HH or HM task.

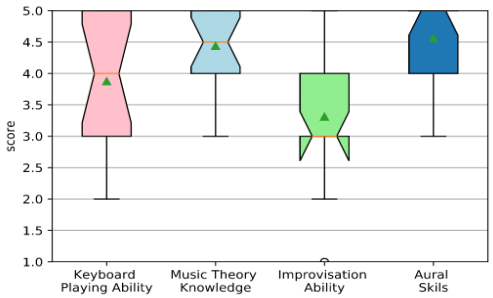

Figure 7: Box plots of the musical background of the 13 participants.

We used the Amazon mTurk platform to recruit 22 participants and we paid them $1 per HIT (Human Intelligent Task) for 6 to 10 minutes of work. Each participant could work on up to 2 HITs. To filter out bots, and workers that have no musical background, we designed a qualification test. The participants were given an image of two measures of complex polyphonic music, and they had to find the number of voices, the key, and the scale degree of a note. Besides the mTurk survey, we created an independent webpage to host the same HIT, in order to make it available for a larger group of people without monetary incentives. Since there was no payment for this task, we did not use the same qualification test as in mTurk, but we asked them to describe their music education level (composer, performer, expert, novice, etc). Out of 44 total participants, only 26 musically trained were selected.

The results of the listening test are similar for both groups of participants (mTurk and our webpage), and indicate that the HM generated duets were at least as good as the HH ones, if not better, as shown in Figure 8. The scores for both categories were almost the same, and even though the HM task achieved a higher score, this difference is not statistically significant (Mann Whitney U test, $a = 0.05$). Furthermore, neither group of participants were able to differentiate HH and HM duets. Specifically, for the mTurk group, 56% of HH duets are classified as HM in aggregation and 47% of HM duets are classified as HH. For the group from our webpage, 53.57% of HH duets are classified as HM and 48.97% of HM duets are classified as HH.

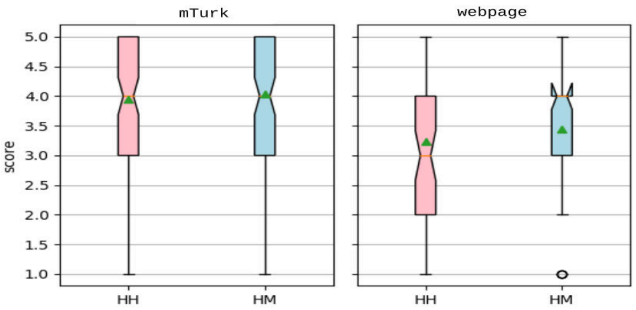

Figure 8: Blind listening test results from both participant groups (mTurk and our webpage).

4.4 Objective Evaluation

In this section, we perform an objective evaluation to further assess the quality of the generated duets. Specifically, we investigate the extent to which the generated duets follow common counterpoint rules. To do so, we developed an algorithm based on music21's *voiceLeading* module, which parses two-voice counterpoint pieces and evaluates each according to several common-practice style guidelines. We consider 9 types of mistakes, each of which is weighted according to its severity based on several music theory textbooks, such as [9]: 1) forbidden parallel: 1, 2) improper resolution: 1, 3) unprepared dissonance: 0.75, 4) hidden fifth: 0.5, 5) hidden octave: 0.5, 6) leap not set with step: 0.5, 7) opens incorrectly: 0.5, 8) melodic dissonance: 0.5, and 9) voice crossing: 0.25. The algorithm iterates through all note pairs that are adjacent within the same voice (i.e., melodic intervals), and stacked vertically between voices (i.e., harmonic intervals), and searches for any of the above mistakes. The final normalized score is the total number of pairs minus the weighted sum of all mistakes, all over the total number of pairs, and it ranges from from 0 (nothing correct) to 1 (no mistakes).

We use the above-mentioned script to evaluate four sets of duets: 1) HM duets used in the listening test, 2) HH duets used in the listening test, 3) duets harmonized by our agent model in an online but non-real-time fashion of a soprano or bass voice from Bach chorales (test set), and 4) soprano-bass duets extracted from Bach chorales (test set). Clearly, Sets 1 and 2 are about real-time interactions between two parties. For Set 3, there is no interaction between the two voices; the agent model simply harmonizes Bach's part. Set 4 serves as a baseline for duet counterpoint.

	HH	HM	Harmonization	Bach duets
score	0.71	0.72	0.76	0.89

Table 1: Objective scores of 4 different types of duets

The above matrix is in accordance with the listening tests, where the perceived quality of both HH and HM duets was very similar. None of the generated duets could achieve a similar score to Bach duets, which is expected, since Bach duets are composed in an offline bidirectional way. This score is just an indicator that our model is adhering to the basic rules of music theory, and by no means, it should be considered a valid descriptor of the the overall musical quality of the duets.

5. CONCLUSIONS

In this work, we proposed *BachDuet*, a novel human-machine interactive system for duet counterpoint improvisation. The machine agent model in BachDuet is a multi-tasking LSTM network. It predicts the token to play by the machine in the next 16th note timestep, as well as the key modulation in the next beat for music notation visualization. User studies suggested that classical musicians showed great interest in duet counterpoint improvisation, and they rated the human-machine interaction experience highly compared to that in human-human interaction. Listening tests and objective evaluations showed that the quality of the generated human-machine duets is comparable to that of human-human duets. For future work, we plan to improve the agent model with more diverse training data and network architecture. Another interesting idea is to extend the system to support acoustic and visual inputs from the user. A computer vision module can be used to interpret the performers' cues and infer the dynamics and tempo.

6. ACKNOWLEDGEMENT

We thank Professor Matthew Brown from the Eastman School of Music at the University of Rochester for fruitful discussions. This work is funded by National Science Foundation grant No. 1846184.

7. REFERENCES

[1] C. Benetatos and Z. Duan. BachDuet: A human-machine duet improvisation system. In *International Society for Music Information Retrieval Conference Late-Breaking/Demo*, 2019.

[2] J. A. Biles et al. GenJam: A genetic algorithm for generating jazz solos. In *Proc. International Computer Music Conference*, volume 94, pages 131–137, 1994.

[3] M. S. Cuthbert and C. Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *Proc. International Society for Music Information Retrieval*, pages 637–642, 2010.

[4] K. Ebcioğlu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3):43–51, 1988.

[5] D. Eck and J. Schmidhuber. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 103:48, 2002.

[6] E. T. Ferand. *Improvisation in Nine Centuries of Western Music: An Anthology with a Historical Introduction*. Arno Volk Verlag, 1961.

[7] N. Jiang, S. Jin, Z. Duan, and C. Zhang. RL-Duet: Online music accompaniment generation using deep reinforcement learning. In *Proc. Association for the Advancement of Artificial Intelligence*, 2020.

[8] A. Joulin and T. Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Proc. Advances in Neural Information Processing Systems*, pages 190–198, 2015.

[9] S. Laitz. *The Complete Musician*. Oxford University Press, 2015.

[10] G. E. Lewis. Too many notes: Computers, complexity and culture in voyager. *Leonardo Music Journal*, pages 33–39, 2000.

[11] A. Mann. *Theory and Practice: The Great Composer as Student and Teacher*. Norton, 1987.

[12] Y. Mann. AI duet. https://experiments.withgoogle.com/ai/ai-duet, 2016.

[13] R. Moore. The decline of improvisation in western art music: An interpretation of change. *International Review of the Aesthetics and Sociology of Music*, pages 61–84, 1992.

[14] F. Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, 2003.

[15] K. Tatar and P. Pasquier. Musical agents: A typology and state of the art towards musical metacreation. *Journal of New Music Research*, 48(1):56–105, 2019.

[16] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[17] I. Xenakis. *Formalized Music: Thought and Mathematics in Composition*. Pendragon Press, 1992.

[18] Y. Yan, E. Lustig, J. VanderStel, and Z. Duan. Part-invariant model for music generation and harmonization. In *Proc. International Society for Music Information Retrieval*, pages 204–210, 2018.