Deep Learning for Music Composition: Generation, Recommendation and Control

Deep Learning for Music Composition: Generation, Recommendation and Control

A dissertation presented

by

Cheng-Zhi Anna Huang

to

The School of Engineering and Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Computer Science

Harvard University

Cambridge, Massachusetts

April 2019

Dissertation Advisors
Douglas Eck and Aaron Courville

Author
Cheng-Zhi Anna Huang

Deep Learning for Music Composition: Generation, Recommendation and Control

Abstract

Technology has always helped expand the range of musical expression, from the fortepiano to synthesizers to electronic sequencers. Could machine learning further extend human creativity? We explore three ways deep learning supports the creative process: generation, recommendation, and control. Generative models can synthesize stylistic idioms, enabling artists to explore a wider palette of possibilities. Recommendation tools can assist artists in curation. Better model control helps artists stay in the creative loop. Furthermore, this control could take place at one or more musically-meaningful levels – the score, the performance, or timbre – or on a non-musical level, such as a subjective quality like "scary." This dissertation posits that deep learning models designed to better match the structure of music can generate, recommend and provide control in the creative process, making music composition more accessible. I describe four projects to support this statement. AdaptiveKnobs (Huang et al., 2014) uses Gaussian Processes to capture the nonlinear multimodal relationship between low-level sound synthesis parameters and perceived sound qualities. By using active learning, we assist sound designers in defining their own intuitive knobs by querying them on sounds that the model expects to improve the controls most. ChordRipple (Huang et al., 2016) uses Chord2Vec to learn chord embeddings for recommending creative substitutions and a Ripple mechanism to propagate changes, allowing novices to compose more adventurous chord progressions. Music Transformer (Huang et al., 2019) uses self-attention mechanisms to capture the self-similarity structure of music, generating coherent expressive piano music from scratch. As the model processes composition and performance as one, improvisers can play an initial motif and have the model develop it in a coherent fashion. Coconet (Huang et al., 2017) uses convolutions to capture pitch and temporal invariance. The generative model fills in arbitrarily-partial musical scores, allowing it to perform a wide range of musical tasks. The model uses Gibbs sampling to approximate how human composers improve their music through rewriting. Recently, Coconet powered the Bach Doodle, harmonizing more than 50 million melodies composed by users. We hope machine learning can enable new ways of approaching the creative process for both novices and musicians.

Contents

Acknowledgements

This work would not have been possible without all of you. I feel very lucky to have found so many mentors and friends who believed in me more than I did, who understood my passion and became more proud of my work then I was. This thesis was a decade-long pursuit, and I feel lucky it was always clear to me that I wanted to make a contribution in how we approach the creative process in music through technology, and that lead me from one place to the next. I have had the fortune to find a home in many institutions, Harvard, MIT, IRCAM, MuseAmi, MILA and Google AI Residency program. Thank you for taking me in and going out of your way to help me grow as a musician and researcher. I feel lucky that I was able to spend time in both the music and computer science department, and to overlap with many generations of students. Seeing how each graduate student invents a new line of research because of the questions and issues they care about constantly inspires me. I am grateful for the support from many communities, from music to ML to HCI, including conferences such as MCM, ICMC, ISMIR, IUI, NeurIPS, ICML and ICLR. I want to give a shoutout to regional meetings such as NEMISIG and NEMCOG for providing a supportive environment to present work-in-progress and to connect with others with shared interest.

First of all, I want to thank Douglas Eck for being there for me every step of the way in the past three years, for always giving me more support and care and insightful technical feedback then I could have ever imagined. Thank you for creating Magenta that allowed me to excel as a researcher in this field.

I want to thank Aaron Courville for taking me on as a student, when I was in between institutions and still trying to find a path forward. Thank you for creating a music generation working group at MILA that gave me the space and time and technical support to improve

Coconet and the theory behind it.

Thank you Krzysztof Gajos for introducing me to the field of HCI and taking the risk to enter a new field. You helped me envision an entire line of research in how we can design systems to better support composers, and I hope to finally be able to realize it some day. Thank you for your patience and generosity in helping me grow as a researcher.

Thank you Avi Pfeffer for advising me and vouching for me even after you left the department. I remember our discussions on music and AI and wish we had more time to work together.

Thank you David Parkes for chairing my committee, being part of my earlier committee meetings, and always offering the most effective advice on how to move forward. Thank you for believing in me, empathizing in the ups and downs, and helping me feel proud in finishing.

Thank you John Girash, Lisa Frazier-Zezze, Patricia Jacome, Julie S. Holbrook, Garth McCavana, Richard McGirr and Jason R. Ortega for helping me navigate the PhD program and for making sure that I had everything in place for a smooth defense.

In addition to my thesis committee, I have been very lucky to work closely with David Duvenaud, Ken Arnold, Josiah W. Oberholtzer, Tim Cooijmans and Ashish Vaswani.

Working with David Duvenaud was a turning point for me and with whom I published my first two papers. Thank you for being an amazing mentor and friend, for helping me through some of the most difficult times in grad school, with your enthusiasm, thoughtfulness and technical expertise. While working with you, I was able to move forward at a speed and technical level I didn't know I was capable of. Thank you for introducing me to Gaussian Processes and the machine learning research community.

Thank you Ken Arnold for being there for me throughout this entire journey of grad school, as a friend, labmate and collaborator. Thank you for always offering to help, from troubleshooting dependencies on music libraries, to interaction design, to brainstorming machine learning models for learning from user feedback. Thank you for regularly reaching out even when I was away from Harvard. I am always inspired by your kindness, generosity and knowledgeability, and I know I can always depend on you for the most clear and insightful feedback.

I want to thank Josiah W. Oberholtzer for also being there since the beginning, for always seeing how my projects could be musically relevant, and always going out of your way to help me succeed. The way you use your technical skills to support your musical aspirations in such diverse ways have been a true inspiration and I hope I can reach your level of fluidity some day.

Thank you Tim Cooijmans for keeping me in good company as I started exploring deep learning and music. Thank you for being my sounding board, for sharing my excitement for Coconet and helping me push it to the next level. It only took three submissions for the paper to be accepted at a conference. Thank you for not giving up and for so much technical intuition. I am proud of how far we have come, that our model powered the Bach Doodle.

Thank you Ashish Vaswani for always offering to help more, as a friend and mentor. Thank you for always listening to me explain the challenges of the domain, and offering brilliant technical advice. Thank you for spending countless hours sketching ideas, pair programming, and debugging presentations, making me a much stronger researcher.

Thank you to all my coauthors, without whom all the work wouldn't have been possible, Jonathan Bragg, Brenton Partridge, Adam Roberts, Jakob Uszkoreit, Ian Simon, Curtis (Fjord) Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, Andriy Stasyuk, Sander Dieleman, Erich Elsen, Jesse Engel, Sherol Chen, Mark Nelson and Daphne Ippolito. Thank you for helping me tirelessly, iterating with me on the story, the math, the code, the designs, the figures, the animations, the demo, the writing, the posters, the slides, the talks and many others. Thank you for being there through many rebuttals, rejections and acceptances.

I want to thank all my friends from Harvard computer science throughout the years, my three twins, Alice Gao, Elena Agapie and Elaine Angelino, friends from MD 138, Varun Kanade, Thomas Steinke, Guille D. Canas, Brendan Juba, Mark Bun, Kai-Min Chung, Scott Lindermann, Jonathan Ullman, Justin Thaler, James Zou, Colin Jia Zheng, Zhenming Liu, Chin Ho Lee, Henry Lam, Yun-Ru Chen.

Thank you to all my friends from MD 217 and IIS, Ken Arnold, Susan Arnold, Pao Siangliulue, Steven Komarov, Chuan-Heng Hsiao, Ofra Amir, Juho Kim, Bernd Huber, Sebastian Gehrmann, Nam Wook Kim, Dimitrios Antos, Edith Law, Katharina Reinecke, Na Li, Sunyoung Kim, Joseph Jay Williams, Tuukka Ruotsalo, Mike Terry, Paola Mariselli, Lahiru Jayatilaka, Keito Uchiyama, Sharon Zhou, Lin Pang and Kevin Tian.

I want to thank all my friends in neighboring offices, both early and later on in my PhD (not in chronological order), Ioannis Gkioulekas, Philip Guo, Andrew Mao, Amy Yu, Mike Ruberry, Bo Waggoner, Uri Braun, Ming Yin, Chien-Ju Ho, Hossein Azari, Jean Pouget-Abadie, Bart van Merriënboer, Thibaut Horel, Anudhyan Boral, Eric Balkanski, Hongyao Ma, Paul Tylkin, Debmalya Mandal, Stacy Wong, Amy Dai, Elif Yamangil, Heather Pon-Barry, Shanying Cui, Michael Miller, Allen Schmaltz, Yoon Kim, Sam Wiseman, David Duvenaud, Melisa Sue Trujillo, Jasper Snoek, Matt Johnson, Diana Cai, Chris Stock, Dougal Maclaurin, Oren Rippel, Michael Gelbart, Andrew Miller, Finale Doshi-Velez, Jose Miguel Hernandez Lobato, Matthew Johnson, Christian Steinruecken, Roger Grosse, Malvika Rao, Jens Witkowski, Haoqi Zhang, Greg Stoddard, Victor Shnayder, David Darais, Daniel King, Emma Alexander, Ben Green, Rediet Abebe, Alan Dunne, Laurence Muller, Bertrand Schneider, Sophia Yakun Shao, Yu Wang, Ariana Minot, Chia-Yung Su, Kevin (Hsieh-Chung) Chen, Chih-Han Yu, Ya'akov (Kobi) Gal, Ece Kamar.

I want to thank my PhD cohort, Tsung-Han Lin, Moritz Baecher, Steve Tarsa, Gregory Malecha, and other friends from SEAS and the GSAS, Elizabeth Smoot, Justin Song, Sui Ann Mao, Amos Tai, Karen Lee, Wallace Hui, Gary Sing, and Vincent Chi Kwan Cheung, Benjamin Sanchez Lengeling, Anna Wang, Yu Lei, Dustin Tran, Mina Pascalito Nassif, the Harvard Argentine Tango Society and Thomas Patrick.

Thank you to fellow CS 181 machine learning class teaching fellows, John Lai, Jeremy Hoon, Panos Toulis and David Parkes who is a very inspiring teacher. Also special thanks to other CS faculty who have helped me along the way, Steve Chong, Margo Seltzer, Sasha Rush, Stuart Shieber, Ryan Adams and Barbara Grosz.

I was fortunate to spend time in the Harvard music department, the Composers' Colloquium and HUSEAC which continues to shape my music thinking. Thank you Josiah W Oberholtzer, Justin Hoke, Gabriele Vanoni, Bert Van Herck, Trevor Baca, Chris Swithinbank, Sabrina Schroeder, Sivan Cohen-Elias, Edgar Barroso, Stefan Prins, Marta Gentilucci, Marek Poliks, Tim Mccormack, Ian Power, Ann Cleare, Ashley Fure, Hillary Zipper, Clara Iannotta, Manuela Meier, Tim Mccormack, Peter McMurray, Victoria Tzotzkova, Michelle Lou, Aaron Einbond, Ean White, Seth Torres, Chaya Czernowin and Steven Takasugi, and Hans Tutschku with whom I have taken many electroacoustic composition classes.

Thank you to many friends from my earlier MIT Media Lab days and MIT at large, Wu-Hsi Li, Yu-hsien Wei, Mihir Sarkar, Joe Diaz, Owen Meyers, Ricardo Garcia Rago, Peggy Chi, Nan-Wei Gong, Jackie Chia-Hsun Lee, Helen Hsieh, Dori Tung-Yun Lin, Yi-Hsiang Chao, Jeremiah Kevin Tu, Dawei Shen, Noah Vawter, Eric Rosenbaum, Peter Alexander Torpey and Timothy J. O'Donnell, and other musicians and friends in Boston, Ryan LaPerle, Brian Friedland, Minghui Lin, Esther Ning Yau, Elaine Lin.

I want to thank the many mentors and friends for keeping me in good company working on music and computing. Thank you Elaine Chew for introducing me to the field and continuing to support me, and also others who were at or visited MuCoaCo, Ching-Hua Chuan, Anja Volk, Isaac Schankler, Christopher Raphael and Gérard Assayag. Thank you Veronika Krausas for formally introducing me to music composition, and Tamar Diesendruck for coming to the rescue on several occasions. Thank you Krzysztof Gajos, Avi Pfeffer, Gérard Assayag, Michael Casey, Hans Tutschku and Steven Takasugi for helping me with grant and fellowship proposals.

I started my PhD at Harvard in Fall 2008. Summer 2009, I jumped on a plane to California. Thank you David Cope and Daniel Brown for the best times at WACM, which was hosted at UCSC in the Redwood forests. Later that summer, I jumped on a bus for 11 hours to Montreal. Thank you Matt Hoffman, Rebecca fiebrink and Eric Nichols at ICMC 2009 for reaching out when I was attending by myself.

Thank you for hosting me during summer 2011 at IRCAM, Gérard Assayag, Philippe Esling, Jean Bresson, Nicolas Donin, Miriam Akkermann, Arshia Cont, Serge Lemouton, Mikhail Malt, Daniele Ghisi, Andrea Agostini, Marta Gentilucci, Andrea Sarto, and at PRISMA, Jacopo Baboni Schilingi, Julien Vincenot, Hans Tutschku, Örjan Sandred, Mikael Laurson and many others.

Thank you NEMISIG and NEMCOG for keeping me in good company from 2010 to 2015 while I was struggling to find my direction, in how to balance my artistic and research goals. Thank you Tristan Jehan, Brian Whitman, Paris Smaragdis, Youngmoo Kim, Michael Casey, Morwaread Mary Farbood, Andy Sarroff, Eric Humphrey, Rachel Bittner, Brian McFee, Finn Upham, Colin Raffel, Andrew McPherson, Katie Kinnaird, Oriol Nieto, Michael Musick, Michael Scott Cuthbert, Psyche Loui and Ichiro Fujinaga.

I presented my first paper during my PhD at IUI 2014, and an earlier workshop version at NeurIPS 2013. Thank you again David Duvenaud, Ken Arnold, Krzysztof Gajos and Josiah W Oberholtzer, and also Tristan Jehan for helping me reach this milestone. At IUI 2014, I met Mark Cartwright and Bryan Pardo, and then at ICMC 2014, Anja Volk, Olivier Lartillot, Dorien Herremans, Kat Agres. Thank you all for much encouragement in the PhD process.

January 2015, I gave a short talk at NEMISIG in Ithaca on Chord2Vec, marking an early milestone for my second paper ChordRipple. Thank you David Duvenaud for introducing me to Word2Vec. Thank you HAMR for giving me the space to prototype my first interface to ChordRipple. Thank you Garth Griffin for braving through the snowstorms to drive us there from Boston. Thank you Doug Turnbull and your family for hosting me. Thank you Eric Humphrey for seeing potential in the project and subsequently inviting me to do an internship at MuseAmi. Thank you Hubert Ho for setting up user studies twice at Northwestern for the ChordRipple project.

Summer 2015, thank you MuseAmi, Robert Taub and Eric Humphrey for carving out space for me to work on research even in a startup. Thank you Leigh Smith my desk neighbor for being a super supportive colleague. I want to thank the music technology labs in NYC for making me feel at home, and also the monthly Spotify hackathons organized by Jonathan

Marmor. Thank you to friends who also spent time in NYC that summer, Andy Sarroff, Paul Tylkin, Dustin Tran, Jean Pouget-Abadie and Bart van Merrienboer. Late summer 2015, thank you MuseAmi for sponsoring me to attend the Deep Learning Summer School in Montreal and to present a poster on ChordRipple. This introduced me to the students there, Tim Cooijmans, Negar Rostamzadeh, Mojtaba Khomami Abadi, Li Yao, Zhouhan Lin, Çağlar Gülçehre, Daniel Jiwoong Im, Orhan Firat and many others. Thank you all for helping me feel at home at UdeM. Also thanks fellow summer school students Jesse Engel and Erich Elsen for helping me navigate the summer school.

Summer 2016, I had a wonderful time interning at Magenta alongside Tim Cooijmans, Kyle Kastner and Natasha Jaques. Thank you Magenta for the best summer and also the later two years when I returned as an AI Resident. Thank you Douglas Eck, Adam Roberts, Curtis Hawthorne, Ian Simon, Monica Dinculescu, Jesse Engel, Fred Bertsch, Sageev Oore, Erich Elsen, Cinjon Resnick, Jialin Song, Andrew Kyle Lampinen, David So, Yingtao Tian, Chris Donahue, Lamtharn (Hanoi) Hantrakul, Carey Radebaugh, Pablo Castro, Daphne Ippolito. I want to thank others in our summer 2015 intern cohort at Google Brain, Laurent Dinh, Vincent Dumoulin, Dzmitry Bahdanau, Junyoung Chung, Ben Poole, Jaan Altosaar, Shane Gu, Stephan Zheng, Aurko Roy, Nicolas Papernot, and others, Rose Yu, David Bieber, David Ha, Colin Raffel, David Dohan, Jasmine Collins, Justin Gilmer, Kelvin Xu, Melody Guan, Sam Schoenholz, Yun Liu, George Dahl, Andrew Dai, Nicolas Boulanger-Lewandowski, Jascha Sohl-dickstein, Mike Schuster and Leslie Phillips. Special thanks to the Google Lunchapella!

Fall 2016 to Spring 2017, I spent a year at MILA in University of Montreal. Thank you Aaron Courville and Linda Peinthiere for making this possible. Thank you friends Kyle Kastner, Johanna Hansen, Tegan Maharaj, David Krueger, Bart van Merrienboer, Laurent Dinh, Marcin Moczulski, Vincent Dumoulin, Gabriel Huang, César Laurent, Nicolas Ballas, Guillaume Alain, Mathieu Germain, Joao Felipe Santos, Chiheb Trabelsi, Rosemary Ke, Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Yaroslav Ganin, Michael Noukhovitch, Olexa Bilaniuk, Shawn Tan, Brady Neal, Chin-Wei Huang, Harm De Vries, Kundan Kumar, Amjad Almahairi, Eugene Vorontsov, Iulian Vlad Serban, Dmitriy Serdyuk, Mohammad Pezeshki, Reyhane Askari, Sina Honari, and many others for making Montreal one of my favourite cities.

Summer 2017 to Summer 2019, I want to thank Magenta and the Google AI Residency program for giving me so much opportunity, freedom and support to work on music and machine learning. The Music Transformer project wouldn't have been possible without many colleagues who have worked on the Transformer and Tensor2Tensor firsthand, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Lukasz Kaiser, Ryan Sepassi, Niki Parmar, Llion Jones, Aurko Roy, Arvind Neelakantan, and many Magenta teammates, Ian Simon, Curtis (Fjord) Hawthorne, Monica Dinculescu, Douglas Eck. Thank you also to co-authors Andrew M. Dai and Matthew D. Hoffman, and many for insightful discussions, Jesse Engel, Adam Roberts, Fred Bertsch, Erich Elsen, Sander Dieleman, Sageev Oore, Carey Radebaugh, Natasha Jaques, Daphne Ippolito, Sherol Chan, Vida Vakilotojar, Dustin Tran, Ben Poole and Tim Cooijmans.

Thank you Mike Mozer for spending hours to help me with my thesis statements. I want to thank Hugo Larochelle, Marc G. Bellemare, Graham Taylor, David Grangier, Martin Wattenberg, Fernanda Viegas, Ron Weiss, Noah Fiedel for feedback at various points of the residency.

Thank you fellow residents Avital Oliver, Benjamin Eysenbach, Carles Gelada Saez, Carlos Riquelme, Debidatta Dwibedi, Ekin Dogus Cubuk, Etienne Pot, Gamaleldin Fathy

Elsayed, Harini Kannan, Ishaan Gulrajani, Jacob Buckman, Jaehoon Lee, Jonas Kemp, Joseph Antognini, Julius Adebayo, Katherine Lee, Kathryn Rough, Kumar Krishna Agrawal, Lechao Xiao, Manoj Kumar Sivaraj, Marcin Lukasz Moczulski, Michael Muelly, Nicolas Ford, Pieter-Jan Kindermans, Prajit Ramachandran, Samuel Laurence Smith, Sara Hooker, Simon Kornblith, Suhani Vora, Supasorn Suwajanakorn, Surya Bhupatiraju, Trinh Hoang Trieu, Xinyang Geng, Yasaman Bahri. Thank you Daniel De Freitas Adiwardana, Gabriel Bender, William Fedus and Wei-Ning Hsu.

Thank you Deanna Chen, Leslie Phillips, Sally Jesmonth, Phing Lee, Dan Nanas, Ruth Alcantara and Natacha Mainville. Thank you Anna Goldie, Azalia Mirhoseini, Been Kim, Hanie Sedghi, Azade Nazi, Golnaz Ghiasi, Timnit Gebru, Mahima Pushkarna, Carrie Cai, Peggy Chi, Jie Ren, Dami Choi and Pi-Chuan Chang.

I want to thank Pie Shop, Diana Huang, Mo Adeleye, Cynthia Le and Catherine McCurry, and Classical Ninjas, Sherol Chen, Jon Liang, Yufeng Guo, Chandra Rajagopal, Taylor Barrella, Justin Zho, David So, Yulan Lin and Derek Wu.

Thank you Katie Kinnaird, Oriol Nieto, and others on the 2018 Joint Workshop on Machine Learning for Music organizing committee for inviting me to give a talk.

Thank you to the Bach Doodle team for making it possible for the world to experience first-hand composing with machine learning: Jacob Howcroft, Leon Hong, James Wexler, Curtis (Fjord) Hawthorne, Adam Roberts, Ann Yuan, Nikhil Thorat, Daniel Smilkov, Lauren Hannah-Murphy, Chris Han, Jennifer Kutz and Jason Freidenfelds.

Thank you to all my friends in Hong Kong from PCMS, all my roommates in the past many years. Many many thanks to my sister for hosting me at her place for two years when I ran out of funding.

I want to thank my parents, Chin Shih Huang and Nancy Hai Yan Huang, my sister Lisa Cheng Ran Huang, my brother Victor Cheng Lai Huang, my grandparents Bin Wang and Wei Da WangZi, and my cousin's entire family for always supporting me in pursuing my aspirations.

Last by not least, I want to thank my partner Tim Cooijmans for sharing my passion and for your infinite support in the past four years.

To my grand parents, parents, siblings, and my partner

Chapter 1

Motivation and Summary

1.1 Motivation

Music composition, like other forms of art, involves a complex creative process. Even though highly varied and personal (Collins, 2016), analysis based on sketches and interviews with both novice and professional composers (Collins, 2005, 2007; Bamberger, 2003; Delalande, 2007) show that there are common patterns that emerge. For example, different phases of the creative process can be characterized as *divergent thinking* or *convergent thinking* (Cropley, 2006). The goal of divergent thinking is to generate as many diverse novel ideas as possible, to some extent "seeking a chance discovery" (Delalande, 2007). Convergent thinking on the other hand could be seen as "seeking the suitable", involving both evaluating the appropriateness of ideas and adapting the few to more closely serve the creative goal (Delalande, 2007). These two processes often interleave multiple times at various stages of the compositional process as musical subproblems arise and are resolved (Collins, 2005).

Both phases can be challenging, for example in the divergent phase we might run into a writer's block in coming up with novel ideas. We may be discouraged to explore more adventurous ideas in the middle of a composition because they involve more additional changes. As we pin down a specific idea to attempt, we transition into the convergent phase but our productivity could be hampered by the need to manually implement all the changes needed to maintain the internal consistency of a piece. This motivates the use of techniques from both machine learning and human-computer interaction so that not only can we leverage computers for its generative capacity to extend beyond the range of possibilities explorable by hand but also to help us work faster by anticipating and automating some of the labor involved in prototyping ideas.

The interpretation of whether computers are assisting a composer in the convergent or divergent thinking phase often depends on how much certainty the composer has in their head and the range of exploration they carry out. For instance say we have a model that can take as input a user's melody and output a harmonization. If the composer considers the melody quite fixed and seeks a harmonization that is characteristic of the genre, then a model trained for this task can directly assist the composer in filling in the suitable chords. In this case the model is used as a support tool for convergent thinking. On the contrary, if the composer is using the tool to explore a wide variety of harmonizations and possibly even changing the melody based on if they liked the output harmonizations or not, then the tool is used in assisting divergent thinking.

We can improve the effectiveness of a tool by considering the behaviors typical in the convergent or divergent phase, and then designing models and interactions to make them easier. For example we can better support divergent thinking by going beyond proposing the most plausible solution, but a wide range of alternatives, allowing users to explore the space and its boundaries. For the melody harmonization example, one can use the same generative model but add a recommendation layer to rank its outputs for relevance.

I propose we can decompose how machine learning assist in the compositional process into three components, *generation*, *recommendation* and *control* to better design for each. Generative models serve as the core building block to generating novel material and also to support downstream creative tasks. Recommendation assists users in navigating this space of possibilities, while control allows users to direct the outcome in ways that best fit their creative needs.

In this thesis, I show by designing deep learning models to better match the structure of music, we can use them to generate, recommend and provide control in the creative process, making music composition more accessible.

In particular, generative models that mimic the self-similarity structure in music can reuse earlier motifs to generate music that sustains minute-long coherence. As humans are not constrained to composing left to right and often revisit previous choices, I adopt generative models that allow for rewriting and show that rewriting improves the quality of the generated music. For recommendation, I draw on the practice of substitutions as a way to explore creative alternatives and show it can help novice composers be more adventurous in choosing chords. Akin to macro knobs that allow users to manually map multiple knobs to one, I show how we can use machine learning to automatically learn these mappings.

1.1.1 Generation

First, we need a rich generative model of the domain in order to lay the foundation to generating novel ideas. This can involve learning the data distribution and then sampling from it. The better the model approximates the data distribution, the more the generated results will resemble the characteristics of the data. As a model learns useful domain features, it can generate examples that are novel but still within the style of the data. This could be useful in the divergent thinking phase to brainstorm ideas and to seek chance discoveries.

In contrast to *unconditioned generation* where the model in the ideal case generates according to the data distribution, *conditioned generation* allows users to provide *context* so that samples can be drawn from a distribution that is the result of taking the context into account. The choice in the level of abstraction of specifying the context can depend on how far along the user is in the compositional process and to what degree the idea is realized in the composer's head. For example, the given context could simply be the genre the musician is intending to write the piece in. The context could be more specific such as a *partial score* where only parts of the music is written out. For example, the context can be an already composed melody, and the model can assist by finding the chords to accompany it.

■ 1.1.2 Recommendation

Second, we propose to use *recommendation* in this generative and creative context so that novel but relevant material do not happen by chance but by design. *Collaborative filtering* is often used to predict how much a specific user may prefer an item based on how other users with similar preferences rated that item. In the creative context, we want to work with novel content as opposed to pre-existing material and preferences are often highly personal.

Conditioned generation allows a recommendation system to generate candidates that are both typical of the genre of music modeled and relevant to what the user is composing. For example in the aforementioned scenario of harmonizing a melody, the system would provide harmonizations that fit with the current user melody. This is often a *structured prediction* problem where not only one chord is predicted but a sequence of chords are inferred so that they work well together along with the given context. A recommendation system can also suggest the user to change part of their melody to make it easier to adopt harmonizations that stretch the possibilities.

In addition to generating candidates, a generative model can also be used to assess how typical a harmonization is, allowing the system to rank the candidates from average to atypical. Depending on whether the composer is in the divergent or convergent thinking phase, they might prefer one end over the other. If the composer has already attempted a solution, and want to seek alternatives, the system can rank the candidates based on similarity to the composer's current solution. Generative models often learn embeddings that can be useful for computing similarity. However, these pre-existing ranking schemes may not align with the goals of the composer whom may want to rank the output based on attributes they want to control.

■ 1.1.3 Control

Third, we want users to be able to control the generative outcome based on attributes they care about. Generation that is conditioned on partial score allows composers to influence the outcome by changing notes in the score, which causes the model to change other notes in the score in order to maintain the internal consistency of the solution. However, for this to be an effective control it relies on users learning this causal relationship. Moreover, users may want to express their requirements on a higher level of abstraction, for example by describing the desired mood of a musical passage as opposed to the notes. With such high-level controls, users can brainstorm music on the level of emotional trajectories. Once a trajectory is composed, a generative model conditioned on this trajectory can prototype a sketch. In the divergent thinking phase, this allows the user to quickly try out different trajectories and hear the realizations. In the convergent thinking phase, such a high-level control can allow the users to quickly adjust non-trivial attributes with less manual effort. To achieve this, we need to establish a mapping from high-level attributes to music that bear those attributes. This mapping will also need to be adaptive in that it can condition on additional musical context so that the generated outcome would be relevant.

■ 1.2 Summary

Through a series of four projects, I illustrate how the components *control*, *recommendation* and *generation* can be investigated independently, and to serve as building blocks in extending the creative process.

In chapter 2, I address the control problem by introducing a new kind of control *knobs* that allow users to directly adjust intuitive qualities of a generated artifacts, bypassing low-level control knobs that are typical of sound synthesis engines (Huang et al., 2014). Using *Gaussian processes*, we designed a novel *active learning* algorithm that optimizes learn this new class of control knobs interactively from human ratings, allowing controls to be personalized.

In chapter 3, I propose to use recommendation to assist novices in composing more adventurous chord progressions (Huang et al., 2016). We built ChordRipple, a creativity support tool that uses learned embeddings such as chord2vec to recommend creative substitutions. To make it easier for users to adopt unusual chord changes, ChordRipple can automatically *rewrite* the current context to maintain internal consistency, providing multi-change recommendations called Ripples in additional to single changes.

In chapter 4, I propose to go beyond modeling music from left to right, but as a rewriting process akin to *Gibbs sampling* (Huang et al., 2017). This is inspired by the fact that writing *counterpoint* is easier than composing from scratch as the former is an iterative conditioned task. For example in counterpoint, the composer might first write a melody or quote one from a prior work. Conditioned on that, the composer writes the other lines, and then conditioned on other lines the melody is being adjusted, and the composer goes back and forth among different parts of the music until all parts are coherent (Fux, 1965). In order to perform Gibbs sampling between all parts of the music, we model all ways of factorization the joint distribution, resulting in a model that is an instance orderless Nade (Uria et al., 2014, 2016). This culminated as Coconet, which composes counterpoint using convolutions, capable of infilling arbitrarily incomplete musical scores.

In chapter 5, I address the challenge of generating music with long-term structure by using *self-attention* based neural networks (Huang et al., 2019) such as Transformers (Vaswani et al., 2017). Prior work focused on modeling 15 seconds ($\sim$ 600 token) of music, using Recurrent Neural Networks which compress history into a fixed size hidden state (Oore et al., 2018). To tackle minute-long sequences, we saw the need for direct access to the past because gaped repetition and self-reference occurs at that timescale. Furthermore, timing, periodicity and relational features are particular important for music, and we found *relative attention* (Shaw et al., 2018), which explicitly modulates attention based on how far apart two tokens are, to work well. In order to scale to sequences of length $\sim$ 3500 tokens, we proposed a new formulation of relative attention that uses a sequence of tensor re-indexing operations. This formulation uses significantly less memory then straightforwardly storing intermediate relative embeddings, from $O(ld)$ to $O(l^2d)$, where l is the length of sequences and d is the hidden size of the network. The result is Music Transformer which is capable of generating minute-long coherence piano music with expressive timing and dynamics.

In the following, I describe the projects in more detail by outlining the problem, the challenges, the technical innovations and the experimental results. I have published all four projects as first-author papers at competitive technical conferences, ranging from the International Conference on Intelligent User Interfaces (IUI), annual conference of the International Society of Music Information Retrieval (ISMIR) and International Conference on Learning Representations (ICLR).

■ 1.2.1 AdaptiveKnobs (Huang et al., 2014, IUI)

Active learning of intuitive control knobs for synthesizers using Gaussian processes

In chapter 2, I show an active learning based approach to learning intuitive personalized control knobs for sound synthesis, allowing users to by-pass low-level controls such as wave-shapes or filter envelopes, and directly adjust high-level attributes of a sound, such as how "scary" or "steady" a sound is perceived to be.

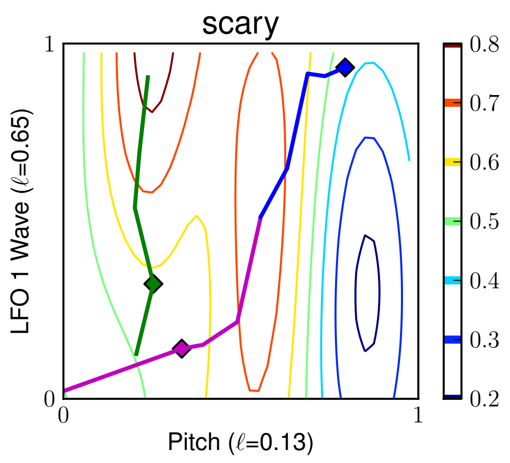


Figure 1.1: A high-level knob such as a "scary" knob consists of paths that are dynamically generated given a set of sounds to be adjusted. The figure shows the mapping from low-level controls (x, y axes) to scariness predicted by model (contour lines). The black diamonds show example sounds that can be adjusted to bear more or less scariness by following the colored paths.

Challenge: High-level Concepts are Multimodal To achieve this, we first model a high-level attribute by using *Gaussian Processes* to learn a function that maps from low-level controls to the perceived levels of that attribute in the sound they create. Figure 1.1 shows the mapping from two low-level control configurations on the x and y axes to the "scariness" of a sound, given by the contour curves. However, the resulting functions are often not one-to-one but multimodal. For example a certain level of "scariness" can be achieved from multiple low-level control configurations, i.e. (x, y) positions in Figure 1.1. The preferred configuration depends on the sound we are adjusting. Ideally, a knob would only adjust the attribute it is supposed to vary and preserve the identity of the sound as much as possible.

New Formulation of Control Knobs Hence, we propose a new formulation for high-level control knobs, that it is not one fixed mapping, but dynamically constructed for each sound, as a path in the low-level control parameters space that follows the gradient of the learned function. For example, starting from the sound configuration itself, the knob can adjust it to sound more or less scary by following the gradient up or down.

Active Learning of Knobs The new formulation of knobs gives us a new objective function that measures the amount of probabiliy mass put on the multiple paths used to adjust a set of starting sounds, i.e. the knob paths in Figure 1.1. By tracking the uncertainty on these dynamically constructed paths, we can use active learning to quickly calibrate the knobs, by querying the user about the sounds the system expects to improve its performance.

Empirical Results We show through simulations that our active learning approach learns high-level knobs faster than several baselines, and that our new formulation of knobs resulted in automatically-constructed knobs that are able to directly adjust non-linear, high-level concepts.

1.2.2 ChordRipple (Huang et al., 2016, IUI)

ChordRipple: Recommending chords to help novice composers go beyond the ordinary

In chapter 3, I introduce a creativity support tool, ChordRipple that uses recommendation to make it easier for composers to experiment with radical chord choices.

Problem Novice composers often find it difficult to go beyond common chord progressions. They are often knowledgeable of the range of possible chords, but find it challenging to find ones that are both interesting and relevant for their own context. Furthermore, adopting an unusual chord change often requires changing other existing chords to make them compatible, which increases the barrier to exploration.

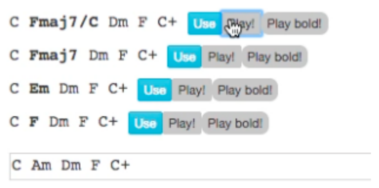

C Fmaj7/C Dm F C+
C Fmaj7 Dm F C+
C Em Dm F C+
C F Dm F C+
C Am Dm F C+

Figure 1.2: An early version of our tool ChordRipple which recommends a diverse range of chords. This particular screenshot shows single chord recommendations for Am, the second chord in the sequence C Am Dm F C+. The bottom two rows show the most common alternatives while the top two row show more adventurous chord choices from chord2vec.

ChordRipple: Assisting Novices by Recommending Chords We propose to use recommendation to assist users in exploring a wider range of relevant chords. We built a creativity support tool, ChordRipple, which makes chord recommendations that aim to be both diverse and appropriate to the current context. Composers can use it to help select the next chord, or to replace sequences of chords in an internally consistent manner.

Adapting Word2Vec as Chord2Vec To make such recommendations, we adapt a neural network model from natural language processing known as word2vec to the music domain, as chord2vec. This model learns chord embeddings from a corpus of chord sequences, placing chords nearby when they are used in similar contexts. We use the learned embeddings to support creative substitutions between chords.

Novel User Interaction: Ripples Changing one part of a sequence often requires additional changes to other parts of the sequence in order to maintain internal consistency. To reduce the overhead of propagating these changes manually, we propose to automatically ripple out these necessary changes and use the entire RIPPLE as a recommendation. These multi-change RIPPLES can make it easier for users to adopt and explore recommendations that are further away from their own.

Empirical Results

Corroboration with Music Theory: Circle-of-Fifths Emerged from Learned Embedding The learned embeddings in CHORD2VEC exhibit topological properties that correspond to theories in music. For example, the major and minor chords are both arranged in the latent space in shapes corresponding to the circle-of-fifths.

Positive Impact on Novice Composers Our structured observations with fourteen music students show that the tool helped them explore a wider palette of chords, and to make "big jumps in just a few chords". It gave them "new ideas of ways to move forward in the piece", not just on a chord-to-chord level but also between phrases. Our controlled studies with nine more music students show that more adventurous chords are adopted when composing with CHORDRIPPLE.

■ 1.2.3 Coconet (Huang et al., 2017, ISMIR)

Coconet: Counterpoint by convolution

Problem Machine learning models of music typically break up the task of composition into a chronological process, composing a piece of music in a single pass from beginning to end. This makes it challenging to support the human compositional process because human composers write music in a nonlinear fashion, scribbling motifs here and there, often requiring inpainting as a form of assistance.

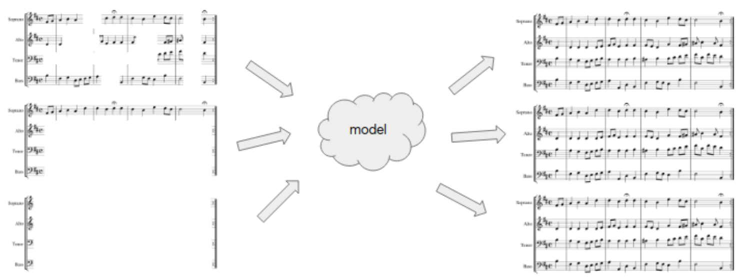


Figure 1.3: COCONET treats different factorizations of music all as an instance of inpainting, allowing a model share parameters among and to be trained jointly on a wide range of musical tasks. Examples of the tasks include, from top to bottom (on the left), partial score completion, harmonization and unconditioned generation.

Orderless NADE: Inpainting as a Generalization of Different Ways of Factorizing Music Inpainting models are trained by masking out parts of the input and then asking the model to learn to fill in through reconstruction. By masking out different parts, the model learns a different way of factorizing music. For example, typical autoregressive models mask out the future, and hence learns the conditions that correspond to factorizing a joint distribution in chronological ordering. To fill in arbitrary partial scores, we need a generative model that is not tied to any particular causal ordering of composing (Figure 1.3). We propose to train an ensemble model that embodies all possible orderings, i.e. $n!$ orderings where n is the number of events. This can be achieved by masking out the input with mask-out size uniformly distributed. We later realized this is an instance of orderless NADE (Uria et al., 2014, 2016).

Gibbs Sampling as an Analogy to Rewriting Composers often improve their compositions through rewriting. Early in the compositional process, much is still influx but as the process progresses more is pinned down. Rewriting allows us to revisit earlier decisions and update them based on this new pinned down information. Similarly Gibbs sampling allows us to leverage this benefit as it iteratively rewrites different parts of a piece conditioned on its complement. Conveniently, an orderless NADE model has all the conditionals possible, making it particularly suitable for Gibbs Sampling.

Using Gibbs Sampling to Turn Unconditioned Generation into Conditioned Generation Furthermore, Gibbs Sampling turns unconditioned generation into a series of simpler conditioned generation problem. Similarly in *counterpoint*, previously written lines become constraints to the current lines, hence reducing the space of possibilities and in many cases making it easier for the composer. As the new lines are composed the older lines are also updated to maintain internal consistency. This process repeats similar to how multiple Gibbs steps are taken.

Counterpoint by Convolution We parameterize our orderless NADE with a convolutional neural network, allowing for time and pitch invariance. Given incomplete scores, it fills in the gaps. We wrap this model in a Gibbs sampling loop to improve its sample quality through rewriting. The result is a generative model we name COCONET.

Empirical results: Both Orderless NADE and Gibbs Sampling Improve Sample Quality COCONET can be used in a wide range of musical tasks, ranging from unconditioned generation, melody harmonization and partial score completion. We evaluate COCONET on the JSB Chorales dataset (Boulanger-Lewandowski et al., 2012). Orderless NADE generalizes better than models that were only trained with chronological ordering. We find that Gibbs sampling greatly improves sample quality, which we demonstrate to be due to some conditional distributions being poorly modeled. Moreover, we show that even the cheap approximate blocked Gibbs procedure from Yao et al. (2014) yields better samples [1] than ancestral sampling, based on both log-likelihood and human evaluation.

1.2.4 Music Transformer (Huang et al., 2019, ICLR)

Music Transformer: Generating music with long-term structure

Problem Music uses repetition and other forms of self reference to build long-term structure. Previous approaches use Recurrent Neural Networks which compress history into a fixed size hidden state (Oore et al., 2018), making it difficult to repeat motifs that are not from the recent past.

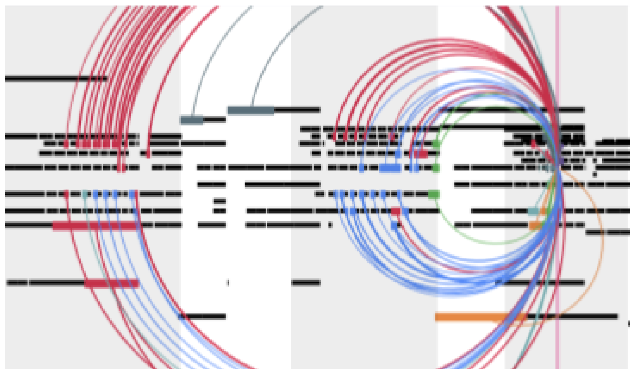


Figure 1.4: Music Transformer generates music from scratch with repeated motives (manually marked by grayed out blocks). The colored arcs visualize self-attention weights and show how previous instances of motifs inform future occurrences, and how the model is able to skip over sections that are less relevant.

Transformer with relative self-attention The Transformer (Vaswani et al., 2017), a sequence model based on self-attention, has achieved compelling results in many generation tasks that require maintaining long-range coherence. This suggests that self-attention might also be well-suited to modeling music, allowing direct access to the entire pass.

However the original Transformer relies on absolute timing signals to keep track of timing and ordering, thus making it harder to learn regularity that is based on relative distances, event orderings and periodicity. Existing approaches for representing relative positional information in the Transformer modulate attention based on pairwise distance (Shaw et al., 2018). This is impractical for long sequences such as musical compositions since their memory complexity for intermediate relative information is $O(l^2 d)$, where l is the sequence length and d is the hidden size of the network.

New memory-efficient formulation of relative self-attention Prior work focused on modeling 15 seconds ($\sim$ 600 token) of music, we aimed to model minute-long music ($\sim$ 2000 to 3500

[1]Samples from Coconet can be heard at https://coconets.github.io/.

tokens). To reduce the intermediate memory requirements, we propose an algorithm that uses a sequence of tensor re-indexing operations, allowing us to by-pass the need to store an intermediate relative embedding for each input token.

Empirical results

Relative attention generalizes beyond the length trained on Transformer with our relative attention formulation is able to generate coherently at lengths longer than it is trained on but the original Transformer deteriorates beyond its training length, perhaps because it is easier for the former to learn relational features.

State-of-the-art results on Maestro dataset and generates music with minute-long coherence Transformer with our relative attention formulation achieved state-of-the-art likelihood on language modeling of the performed piano MIDI tracks in the Maestro dataset (Hawthorne et al., 2019). In listening tests, samples from Transformer with our relative attention was rated statistically significantly higher then samples from the original Transformer. Our model was able to generate minute-long compositions (thousands of steps, four times the length modeled in Oore et al. (2018)) with compelling structure and generate continuations that coherently elaborate on a given motif[2].

[2]Samples from Music Transformer are available for listening at http://g.co/magenta/music-transformer

Chapter 2

AdaptiveKnobs: Active Learning of Intuitive Control Knobs for Synthesizers Using Gaussian Processes

Typical synthesizers only provide controls to the low-level parameters of sound-synthesis, such as wave-shapes or filter envelopes. In contrast, composers often want to adjust and express higher-level qualities, such as how ‘scary’ or ‘steady’ sounds are perceived to be.

We develop a system which allows users to directly control abstract, high-level qualities of sounds. To do this, our system learns functions that map from synthesizer control settings to perceived levels of high-level qualities. Given these functions, our system can generate high-level knobs that directly adjust sounds to have more or less of those qualities. We model the functions mapping from control-parameters to the degree of each high-level quality using Gaussian processes, a nonparametric Bayesian model. These models can adjust to the complexity of the function being learned, account for nonlinear interaction between control-parameters, and allow us to characterize the uncertainty about the functions being learned.

By tracking uncertainty about the functions being learned, we can use active learning to quickly calibrate the tool, by querying the user about the sounds the system expects to most improve its performance. We show through simulations that this model-based active learning approach learns high-level knobs on certain classes of target concepts faster than several baselines, and give examples of the resulting automatically-constructed knobs which adjust levels of non-linear, high-level concepts.

■ 2.1 Introduction

Composers and sound designers use synthesizers to create sounds with spectral and temporal dynamics beyond what is possible through acoustic instruments or recorded sounds. These users generally seek not just to create a certain sound, but rather a collection of related sounds, by transforming a set of sounds in semantically meaningful directions. The ability to intuitively adjust only the relevant qualities of a set of sounds is paramount to the artist's workflow. However, synthesizers are particularly difficult for artists to interact with, since their many parameters must often be adjusted in complex ways in order to change just a single semantically-meaningful aspect of a sound. This often forces a shift in the user's attention from expressing their high-level goals to the low-level trial-and-error tweaking of control parameters.

Several strategies have been developed to address this problem: First, providing the user with pre-set sounds having various qualities, such as "bright acoustic piano" or "dark acoustic piano". Second, interpolating between existing sounds, by averaging the low-level control settings which generate those sounds. Third, expert-engineered high-level knobs for directly controlling relatively intuitive aspects of sounds, such as Waves' OneKnobs, mostly used in the context of production to "brighten" or "phatten" tracks and mixes, and which under the hood controls sound-treatment modules such as equalizers, compressors, and resonators.

However, synthesizers can support more complex concepts, such as those that are multi-modal. For example there are different kinds of "scary" sounds, such as low rumbling, cold chilly wind-like sounds. And we might want different ways to adjust the "scariness" of a sound depending on what kind it is. For example, if a sound is directional, we might make it more "scary" by making it more aggressive. If a sound is more of an ambient pulsating sound, we can make it warble in more irregular ways or more high pitched, or perhaps add some eerie metallic "clicks" to its foreground.

In this paper, we explicitly model high-level sound qualities in order to allow users to automatically adjust arbitrary sounds to have desired levels of those qualities. We define a high-level concept as a function that maps from a high-dimensional control space to the perceived level of that quality. With such mappings, we can generate a high-level knob which directly adjusts sounds to have more or less of a given quality. These knobs are generated for each sound by constructing a path through control space which increases (or decreases) the level of the learned function. Such paths allow composers to move towards and away from sounds along perceptually meaningful dimensions. To learn such functions for a high-level concept, we learn from user ratings of sounds generated by the synthesizer.

In order to learn from a small number of user interactions, we take an active learning approach to assisting a user in finding points in the control-parameter space to demonstrate a high-level concept. The user first informs the system which sorts of sounds she wants to be able to modify and to which ranges. At any point in teaching the system a high-level concept, the user can ask the system where it wants to learn about the high-level concept the most. The system queries the user on the sound that it believes can most improve its ability to adjust the high-level quality for the given set of sounds. The user can listen to the sound and respond by rating how much they think the sound has the high-level quality. We have prototyped a user interface to allow users to express this rating by allowing them to organize sounds on a continuous one-dimensional space to indicate how much they think the sounds carry a particular high-level quality.

Our formulation can be applied to other domains where a user wants to build a layer of richer, personalized controls on top of the parameters provided by the original system.

■ 2.2 Related work

We describe related work in computer-assisted sound design and music composition that uses the metaphor of knobs (one-dimensional, continuous controls), and also other kinds of interfaces and interactions that assist users in working with synthesizers. Our active-learning approach is informed by work in active learning in general, and active data selection, and Bayesian optimization.

■ 2.2.1 High-level knobs

There have been a number of works on learning high-level "knobs" for audio editing tools such as equalizers and reverberators. Many treat a knob as a fixed linear mapping between a low-level control and a high-level quality (Sabin and Pardo, 2009; Sabin et al., 2011; Pardo et al., 2012) . They adopt a weighting-function procedure widely used in psychoacoustics to tune hearing aids, which operates by correlating gain in each frequency bin to a user's perceptual ratings.

Instead of mapping directly between parameters of a specific reverberator and user ratings, Rafii and Pardo (2009) derived nonlinear mappings from low-level controls to reverberation measures such as "echoness", which can be generalized onto different reverberators with similar parameterizations. Their system then learns linear mappings from reverberation measures to perceptual qualities such as "boomy". As sound modification on synthesizers is more complex and their parameterizations vary drastically across different synthesizers, it is more difficult to engineer a set of high-level functions *a priori*. Instead, we use the expressiveness of nonparametric methods to model the covariance structure between controls and to adjust to the complexity of functions being learned.

As user ratings for different high-level qualities may be correlated, Rafii and Pardo (2009) uses transfer learning to incorporate concepts taught by prior users. In this setting, each user rates only a subset of example sounds to train a user-concept, and the system fills in the rest of the ratings by weighting ratings of other user-concepts by their distances to the current user-concept. To choose the subset of sounds to query a user, the work uses active learning by selecting sounds that most differentiate between previously learned user-concepts. Cartwright and Pardo (2013) crowd-sources many more user-concepts to study which equalizer descriptors are widely-agreed-upon and which are true audio synonyms. Our method takes an active-learning approach to help a user more quickly teach a concept from scratch, with the objective of maximizing test-time performance, defined as the ability to construct high-level knobs that can adjust desired qualities in sounds with high certainty.

In additional to the "knobs" metaphor for interaction, Mecklenburg and Loviscach (2006) uses a 2D self-organizing map to place concepts with similar ratings close to each other, so that users can directly explore the semantic space instead of control space. Fiebrink et al. (2009b) allows users to interact with different supervised learning algorithms to define mappings from arbitrary controllers to synthesis parameters. Fiebrink et al. (2009a) describes a play-along mode where the user first composes a sequence of synthesis parameters and then gestures along as it is being played back. The parameter-gesture pairs become training data for learning a mapping that supports the reversed interaction. Instead of relying on user data, Loviscach (2008) applies data mining to existing presets to provide autocompletion. For example, as a user adjusts the controls, other statistically related controls adjusts itself accordingly.

A complementary interaction is when the user is not searching for a desired sound, but already has an audio recording of a target sound at hand, and wants the machine to automatically generate a synthesizer setting that approximates that sound. With such a setting, the sound is no longer canned, but can be manipulated in real-time using the synthesizer controllers. This is often achieved by minimizing the difference between the timbral trajectories of the synthesized and the target sound. As synthesizer controls are high dimensional and highly nonlinear, optimization techniques such as genetic algorithms or particle swarm optimization are often used to search through different synthesis structures and to perform parameter estimation (Garcia, 2002; Heise et al., 2009; Yee-King, 2011; Macret et al., 2012). These techniques also allow users to control a synthesizer by directly specifying values for acoustic features (Hoffman and Cook, 2006).

The "knobs" metaphor is also used in computer-assisted composition for helping users adjust high-level qualities of symbolic representations of music. For example, Morris et al. (2008) exposes the log weighting between transition matrices in hidden Markov models trained on major ("happy") and minor ("sad") songs, as an intuitive knob for adjusting how "happy" an accompaniment sounds. Pachet (2009) adjusts melodies to be more or less 'tonal', 'serial', or 'brown' by moving closer or further away from that concept's decision boundary.

■ 2.2.2 Active learning

A number of systems have explored using active learning for assisting users in defining personalized concepts (Fogarty et al., 2008; Amershi et al., 2011; Pardo et al., 2012). For example, Fogarty et al. (2008) supports users in interactively defining concepts for re-ranking web images search results, and presents users the option of classifying images that are closest to the decision boundary between positive and negative classes, using a nearest-neighbor approach. They also use a heuristic to actively select images that explore new weightings in their distance metrics on low-level features used in computer vision.

More generally, one of the goals of active data selection is to select data that reduces the uncertainty the most on some parameters of interest, given past observations. For example, we can aim to maximize the expected information gain on model parameters by selecting data where the the posterior predictive variance is the highest (MacKay, 1992). Alternatively, the goal could be to minimize generalization error by choosing data that minimizes the overall variance of the predictor (Cohn). Both of these criteria have been employed in active Gaussian process GP regression, by leveraging the variance on the posterior predictive marginal distribution of GPs (Seo et al., 2000). Kapoor et al. (2007) adapts this approach into a classification setting of visual object category recognition, defines a covariance function that reflects local features for object and image representations, and then actively selects data closest to the decision boundary.

However, for settings where we are not given a set of examples a priori to select from, we need to first propose a set of queries from a continuous space, which is the case for our work. Furthermore, the objective may be more specific than reducing uncertainty on parameters. More generally, active learning is the setting where a model and an associated goodness (or loss) is first defined and then the goal is to query a next point that is predicted to result in a future model that produces the highest goodness (or equivalently lowest loss).

In our work, we define a new utility function for our domain by formalizing what the user would desire during test time. Our utility function captures how confident a model is in producing the desired knob paths for adjusting a given set of starting sounds. As in Osborne et al. (2010) and Brochu et al. (2010), our system learns in an iterative manner, and at each iteration we query the user with the point that maximizes the tool's expected utility. While Brochu et al. (2010) uses Bayesian optimization to search globally to help users in procedural animation to find the best parameter setting, our goal is to focus learning in regions that are expected to later allow us to best fulfill user requests. As in Snoek et al. (2012), we compute the expected utility of a point by sampling its values from the current distribution, and evaluate the model's expected confidence in adjusting sounds, conditioned on fantasized ratings.

■ 2.3 Overview of method

We treat a high-level concept as a function that maps from synthesizer control parameter space $\mathbf{x} \in \mathbb{R}^D$ to perceived $y \in \mathbb{R}$ levels of that quality. If we knew this function $f(\mathbf{x})$, we could automatically adjust that concept on a preset $\mathbf{x}_s$ to a desired level y_d of that quality by moving $\mathbf{x}_s$ to a nearby location that has the desired quality level. A knob would move $\mathbf{x}$ along a continuous path through control-parameter space corresponding to increasing and decreasing levels of $f(\mathbf{x})$.

Since we do not know $f(\mathbf{x})$, we start with a probabilistic model of $f(\mathbf{x})$ and refine it with user feedback. Because the domain of f will be high-dimensional, learning the function over its whole domain would require an impractical amount of user feedback. Instead, we focus our learning in high-level knob space by asking for user feedback only in the parts of the control parameter space that is relevant to improving the paths for the high-level knobs.

To guide the acquisition of user feedback, we first specify how we expect the system will be used (which sorts of sounds will be modified, and the expected range of modifications). We can then estimate how well the tool can be expected to fulfill the user's request, given the different queries made to the user during training time. We can then ask the user for the feedback which is most expected to improve the test-time performance of the system. In short, we will use active learning to select the queries made to the user in order to determine f in places which will be expected to improve the utility of the tool the most.

Determining good user queries requires two steps: first, proposing a set of possible user queries, then ranking those queries by how much they are expected to improve the utility of the tool. For the former, we develop a heuristic which proposes points nearby points visited during path optimizations. Second, to estimate a candidate's expected impact of the utility of the system, we sample from the current posterior marginal distribution of f and, for each sample, evaluate the utility of the system as if we had actually made that observation. The candidate point that maximizes for expected utility of the system is chosen for user feedback.

2.4 Modeling high-level knobs

We model the function f mapping from control parameters to high-level concepts probabilistically with Gaussian process (GP) priors (Rasmussen and Williams, 2006). In contrast to previous approaches which learned linear mappings (Pardo et al., 2012), the non-parametric nature of GPs allows us to learn a function whose complexity can grow to match that of the function being learned. We use a zero-mean prior and the standard squared-exponential kernel:

$$\begin{aligned} f &\sim GP(0, k) \\ k(\mathbf{x}, \mathbf{x}') &= \sigma_y^2 \exp(-\frac{(\mathbf{x}-\mathbf{x}')^2}{2\ell^2}) + \sigma_n^2 \delta_{ii'} \end{aligned}$$

The kernel is parameterized by lengthscales ℓ, which specify the typical distances along which each dimension of f varies, and by the amplitude and noise variances σ_y^2 andσ_n^2, respectively. We denote these parameters collectively as θ.

GP regression is an appropriate tool for this problem for two reasons: First, it provides everywhere a closed-form estimate of the remaining uncertainty about the function being learned, which is necessary for estimating the expected performance of the model. Second, the marginal likelihood gives us a principled way to choose the parameters of the kernel.

2.4.1 Integrating over hyperparameters

Typically, the parameters of the kernel are estimated by maximum likelihood. However, when there are very few datapoints, these point estimates are known to drastically over-estimate lengthscales, leading to an under-estimate of uncertainty about the function itself. As we need a useful estimate of uncertainty even conditioned on only a few data points, we must also characterize our uncertainty about the lengthscales of the GP. Therefore we take a fully-Bayesian approach by approximately integrating over hyperparameters using a Sobol sequence.

Our approximate integration over kernel parameters means that our predictive marginal posterior of f at a given point $\mathbf{x}^\star$ is a weighted sum of GP posteriors:

$$\begin{aligned} P(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) &= \int P(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}, \theta) P(\theta|\mathbf{X}, \mathbf{y}) d\theta \\ P(\theta|\mathbf{X}, \mathbf{y}) &= \frac{P(\mathbf{y}|\mathbf{X}, \theta) P(\theta)}{\int P(\mathbf{y}|\mathbf{X}, \theta') P(\theta') d\theta'} \end{aligned}$$

The mean and variance of the posterior marginal at $\mathbf{x}^*$ is given below. $\mu_i(\mathbf{x}^*)$ and $\sigma_i^2(\mathbf{x}^*)$ are the posterior mean and variance of the GP with a single set of hyperparameters θ_i.

$$\begin{aligned} \mu_i(\mathbf{x}^*) &= \mathbf{k}_{\theta_i}(\mathbf{X}, \mathbf{x})^T \mathbf{K}_{\theta_i}^{-1}(\mathbf{X}, \mathbf{X}) \mathbf{y} \\ \sigma_i^2(\mathbf{x}^*) &= \mathbf{k}_{\theta_i}(\mathbf{x}, \mathbf{x}) - \mathbf{k}_{\theta_i}(\mathbf{X}, \mathbf{x})^T \mathbf{K}_{\theta_i}^{-1} \mathbf{k}_{\theta_i}(\mathbf{X}, \mathbf{x}) \\ \mu(\mathbf{x}^*) &= \sum_{i=1}^{M} P(\theta_i|\mathbf{X}, \mathbf{y}) \mu_i(\mathbf{x}^*) \\ \sigma^2(\mathbf{x}^*) &= \sum_{i=1}^{M} P(\theta_i|\mathbf{X}, \mathbf{y}) \left([\mu_i(\mathbf{x}^*) - \mu(\mathbf{x}^*)]^2 + \sigma_i^2(\mathbf{x}^*) \right) \end{aligned}$$

To sample a y^* given x^*, we first sample which set of hyperparameters to use, and then sample from the predictive marginal posterior given just that set of hyperparameters.

■ 2.4.2 Knob paths

Once we have an estimate of the function defining a high-level quality, we can generate a knob which varies that quality for a preset sound by finding a path from the preset through control-parameter space corresponding to increasing and decreasing levels of the learned function. We generate such paths by first locally optimizing for each of a set of equally-spaced desired quality levels $\mathbf{y}_d$ spanning from the lowest user rating to the highest, then linearly interpolate between these points. In our experiments, we used eight equally spaced levels. The objective of this optimization is to minimize the squared difference between the posterior predictive mean and a desired knob level y_d when starting from a control-parameter setting $\mathbf{x}_s$. We denote the control-parameter setting returned by the optimizer as $\mathbf{x}_{sd}$.

$$\mathbf{x}_{sd} = \arg\min_{\mathbf{x}^*} \left(y_d - \mu(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y})\right)^2 \tag{2.1}$$

In future work, we plan to take uncertainty into account while optimizing for desired knob levels, and explicitly optimizing for desirable properties for knobs such as smoothness.

2.5 Active learning

Our system learns in an iterative manner: At each iteration, we choose a point at which to query the user, based on the current model of f. We then update the model of f, and re-query the user. To determine which point to query the user at, we first propose a set of *candidate points*, whose expected impact on the utility of the system we will estimate. After estimating these utilities, we then query the user at the best candidate point.

2.5.1 Evaluating the expected utility of a point

Given a set of *candidate points*, we need to decide which one can most help us improve the utility of our model. Formally, we define *utility* $\mathcal{U}$ as the total mass within ϵ-wide bins centered at the desired knob levels $\mathbf{y}_d$ for all presets $\mathbf{X}_s$.

$$\begin{aligned}\mathcal{U}(\mathbf{X},\mathbf{y}) = & \sum_{s=1}^{S}\sum_{d=1}^{L}[P(y_{sd} > (y_d+\epsilon)|\mathbf{x}_{sd},\mathbf{X},\mathbf{y}) \\ & - P(y_{sd} > (y_d-\epsilon)|\mathbf{x}_{sd},\mathbf{X},\mathbf{y})]\end{aligned} \tag{2.2}$$

This expression can be intuitively thought of as the probability that we will be able to give the user a sound with the desired high-level quality by moving along the learned paths.

2.5.2 Reoptimizing hyperparameters under fantasies

To evaluate the expected impact of a candidate $\mathbf{x}_c$ on our system, we sample the current posterior predictive marginal distribution $P(y_c|\mathbf{x}_c,\mathbf{X},\mathbf{y})$ for what the possible perceived y_c could be. In order to more accurately evaluate expected utilities, we retrain our model for each sample by re-optimizing the hyperparameters on the GP prior as if we had seen this additional observation, resulting in a fantasized posterior predictive marginal distribution of $P(y^*|\mathbf{x}^*,\mathbf{x}_c,y_c,\mathbf{X},\mathbf{y})$ on which the desired knob levels are optimized. Intuitively, the hyperparameter re-optimization is helpful because it accounts for the fact that new queries can potentially change our estimates of the lengthscales, which can drastically change our model's predictions. This is essential early on when we have not yet seen much data, and are very uncertain about the lengthscales.

$$\mathbf{E}(\mathcal{U}|\mathbf{x}_c,\mathbf{X},\mathbf{y}) = \int \mathcal{U}(\mathbf{x}_c,y_c,\mathbf{X},\mathbf{y})P(y_c|\mathbf{x}_c,\mathbf{X},\mathbf{y})\mathrm{d}y_c \tag{2.3}$$

2.5.3 Proposing candidate points

Intuitively, we want to learn more about points that can potentially improve the knob paths. We use a heuristic for generating candidate points which are likely to be useful: We sample points nearby those visited by the truncated-Newton optimizer while determining the knob paths.

Because we expect f to vary slowly in directions which have long lengthscales, we heuristically add gaussian noise to candidates $\mathbf{X}_c$ with variances proportional to the length scales of that dimension. Hence, to determine the noise for a particular candidate x_c, we first sample a set of length scales according to their marginal likelihood. Then, for each dimension d, we sample from a gaussian with variance proportional to the lengthscale l^d of that dimension. We reject any points that fall outside the domain of the synthesizer.

$$\epsilon_c^d \sim \mathcal{N}(0, \frac{l^d}{2})$$
$$\mathbf{x}_c^d \leftarrow \mathbf{x}_c^d + \epsilon_c^d$$

To further increase the potential information gain, we also heuristically include as candidate points the peaks on the posterior variance of f that are reachable through the truncated optimizer from the given set of presets $\mathbf{X}_s$.

■ 2.5.4 The active-learning algorithm and its variations

Algorithm 1 outlines our procedure for how to decide where the query the user next for feedback. Both `for` loops can be parallelized. Figure 2.1 shows a synthetic 1D visualization of some the steps involved.

Algorithm 1 Choosing the next point $\mathbf{x_r}$ for user to rate

```
**Input:** $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, $\mathbf{X}_s$, $\mathbf{y}_d$, $n_c$: num of candidates, $n_m$: num of monte carlo samples
Train full-bayes $GP$ with $\mathcal{D}$
Optimize (2.1) for $\mathbf{X}_{sd}^{\text{path}}$ on $GP$ given $\mathbf{y}_d$ and $\mathbf{X}_s$
Randomly choose $n_c$ candidates $\mathbf{X}_c$ from points visited during previous optimization step
for $i = 1$ to $n_c$ do
    for $j = 1$ to $n_m$ do
        Sample $y_{c_{ij}} \sim P(y|\mathbf{X}_{c_i}, \mathcal{D})$
        Fantasize $GP_{c_i}$ by training it with $y_{c_{ij}}$, $\mathbf{X}_{c_i}$, $\mathcal{D}$
        Optimize (2.1) for $\mathbf{X}_{sd_{ij}}^{\text{path}}$ on $GP_{c_i}$ given $\mathbf{y}_d$ and $\mathbf{X}_s$
        Compute $\mathcal{U}_{ij}$ by (2.2)
    end for
    $\mathbf{E}(\mathcal{U}_i) = \frac{1}{n_m}\sum_{j=1}^{n_m}\mathcal{U}_{ij}$
end for
maxIndex = $\arg\max \mathbf{E}(\mathcal{U}_i)$
$\mathbf{x}_r = \mathbf{X}_{c_{\text{maxIndex}}}$
```

We later refer to our full-fledged path-informed model-based active-learning procedure as “active learning (re-opt)”. We also experimented with a number of simplifications, both as baselines to compare against, and also because they can have shorter run times and so can be more naturally integrated into interactive systems. “Active learning” takes out the step of reoptimizing hyperparameters after each fantasized outcome. “Path entropy” skips the fantasizing step all together, and chooses the point that has the highest variance from the set of proposed candidate points, while “Path random” simply chooses a point at random from the proposed points.

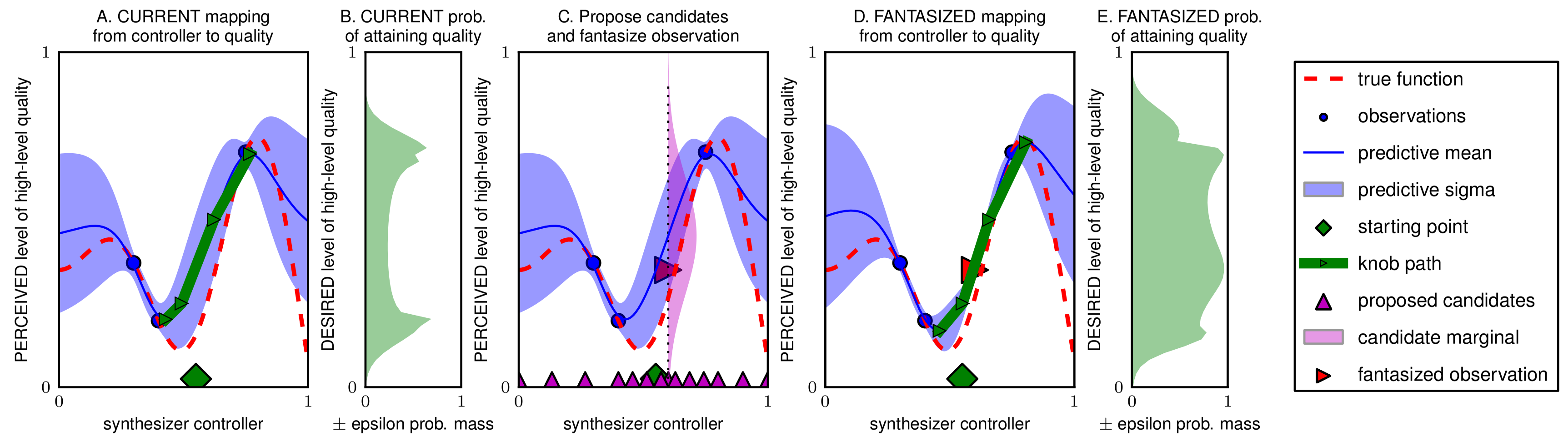


Figure 2.1: Synthetic 1D example of an 1-sample evaluation of the expected utility of a proposed candidate, where the model's utility increased from (B) to (E). Plots A, C, D refer to mappings from control-parameter (x-axis) to a high-level quality (y-axis). Plots B, E refer to the model's confidence in attaining desired levels of a high-level quality (y-axis) measured by the$\pm\epsilon$ probability mass (x-axis) from the mean of the predictive marginal for the corresponding control-parameter on the knob path (green line in preceding plot). (A) Current mapping from synth controller to high-level quality, given three ratings of perceived levels of high-level quality. The green diamond is a sound in the synth control space that is to be adjusted. (C) Proposed candidates shown as magenta triangles on x-axis. The Gaussian predictive marginal of one of the candidates is shown and the red-triangle shows a sample from that distribution. (D) The mapping after fitting the model with the addition fantasized observation (red-triangle).

2.6 User interface and interaction

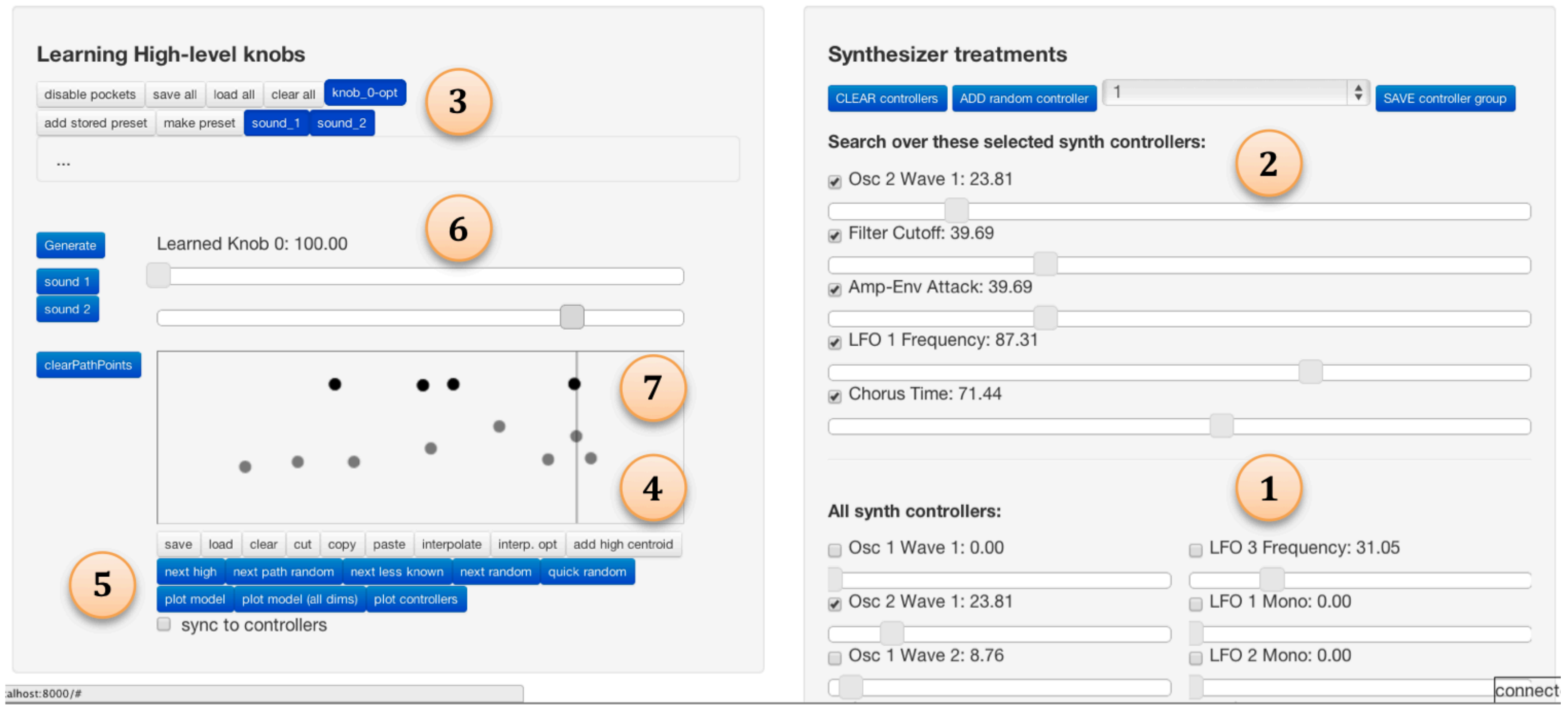
In order to support users in defining their own high-level concepts, we have prototyped a user interface that allows users to demonstrate high-level concepts with examples. The user can communicate how much a sound carries a high-level concept by placing it accordingly into a box, as shown in Figure 2.2, where the horizontal axis indicates an increasing level of a high-level concept from left to right. This allows users to rate sounds in reference to each other, to correct previous ratings, and also to re-adjust their overall scales.

If the user has some prior knowledge of which control parameters might give rise to a high-level concept, she can help the machine reduce the dimensionality of the problem by selecting a subset of parameters, as illustrated by labels 1 and 2 in Figure 2.2. The user can also make such decisions later on in the process as she gains more experience with the synthesizer.

As the user has accumulated a number of points in the box, the user can ask the machine to learn the concept and apply it to a starting sound in order to check how well the machine is understanding the concept. This can be performed at the interface by first hitting the "play" button, at which point the system trains the model and then optimizes for a path through the sound to increasing and decreasing levels of the learned concept. The system has been set to optimize for three discrete levels on the learned function. Both these four points and the original sound are added to the box and placed horizontally at where the model believes it should be. These points are the "darker" row of dots labeled as 7 in Figure 2.2, where the second dot from the left is the original sound. The user then has the option of correcting these points by moving them left or right, and the moved points become a new data point for training the high-level concept. The user can now move the slider above the box to adjust her sound according to the learned concept, and see how it controls the synth control parameters.

2.6.1 Assisting the user with active learning

The user can specify how she expects to use the system by first choosing a set of presets she wishes to modify. This can be indicated in the interface by suppressing those presets, labeled as 3 in Figure 2.2. Then at each iteration, the user can ask for different kinds of assistance by clicking on the corresponding helper buttons. For example, the "next filler random" button labeled as 5 calls for the "path random" variation of our active-learning algorithm to propose the next point to evaluate. A new control parameter setting is then added as a dot to the box, placed at a horizontal position where the model currently believes how much of the high-level concept it carries. The user can help refine the model by correcting these points.



Learned Knob 0: 100.00

Search over these selected synth controllers:

- [✓] Osc 2 Wave 1: 23.81
- [✓] Filter Cutoff: 39.69
- [✓] Amp-Env Attack: 39.69
- [✓] LFO 1 Frequency: 87.31
- [✓] Chorus Time: 71.44

All synth controllers:

- [ ] Osc 1 Wave 1: 0.00
- [✓] Osc 2 Wave 1: 23.81
- [ ] Osc 1 Wave 2: 8.76
- [ ] LFO 3 Frequency: 31.05
- [ ] LFO 1 Mono: 0.00
- [ ] LFO 2 Mono: 0.00

Figure 2.2: User interface for learning and applying high-level knobs to sounds. (1) Sliders for adjusting the low-level control-parameters on the synthesizer directly. There are a total of 64, many are cropped in this screenshot for space. Users can choose a subset of low-level controls to optimize over by checking the checkboxes, and these controls will pop up in region (2). (3) Preset buttons for users to choose as starting sounds. Users can also instantiate new sounds as buttons. (4) Box for users to give ratings to sounds. The horizontal axis of the box reflects how much a sound carries a high-level concept, increasing from left to right. Each training example is represented as a dot and corresponds to a particular synthesizer control-parameter setting. Users can click on buttons in region (5) to ask the tool to suggest sounds to add as training examples. (6) Sliders for adjusting high-level qualities in sounds directly once the model is trained. They are like macro knobs that control the low-level synthesizer parameters in region (2). Each slider corresponds to a knob path that is specific to a starting preset in region (3). (7) The row of darker dots correspond to optimized points on the high-level knob path.

2.7 Experiments and evaluations

We ran a preliminary pilot study with two composers to collect high-level concepts. We evaluate our method in two parts: how well our model captures these high-level concepts, and how quickly our active-learning approach is able to learn these high-level knobs compared to other baselines. For each concept, we show a few examples of knob paths.

2.7.1 Pilot study with two composers

The task was for users to build a knob for a high-level quality such as "scariness" that she wishes to more directly control during sound synthesis. First, the user was prompted to choose a set of preset sounds on which she would test the high-level knob. The user was then asked to identify a set of examples on the synthesizer that carries varying degrees of that high-level quality. Users interacted with our interface shown in Figure 2.2, with the option of asking for suggested examples from the lightweight variation "path random" of our active-learning. The interface allows users to specify the amount of a perceived quality in a sound by moving the dot that represents it on a one-dimensional axis. The synthesizer used for the user studies is a representative software synth named FreeAlpha, from Linplug.

2.7.2 Modeling high-level concepts

From these two pilot studies, we collected three examples of nonlinear concepts on synthesizers, "pulsation", "guitar-like" and "scary". The first and third are multimodal, while the second is unimodal. They range from concrete to abstract. The "pulsation" concept was obtained by querying a single user uniformly on a 8x8 grid in the control-parameter space of 2 synth controls. As we wanted to use this concept for our later simulations, we did not want to bias the input distribution with any of the methods that were being compared in the simulations. For the "guitar-like" and "scary", the user was free to interact with our interface and ask for suggested queries from the "path-random" variation of our active-learning method.

We first illustrate these concepts and then show cross-validation results on how our model performs when trying to predict how users would rate a sound.

Example concept 1: "pulsation"

We want to learn about how two control parameters, the rate of low frequency modulation FM and amplitude modulation AM, interact to produce different degrees of perceived "pulsation". Turning either of these controls up increases pulsation up to a point where modulation becomes so fast that we can not perceive distinctive vibratos anymore, but instead we hear a timbral color change. Moreover, they interact in how they give rise to perceived "pulsation". When FM is low, it dominates our perception, and turning up AM does not give any effect. However when FM is so high that we cannot hear the pulsation anymore, for example the diamond in the lower right corner of Figure 2.7, turning AM up in the lower ranges allows us to add more "pulsation" to our colored sound. We can observe this effect on the response surface of the mean of the GP conditioned on a grid of 8x8 user ratings on these two control dimensions, shown in Figure 2.7.

Example concept 2: "guitar-like"

Figure 2.3, left, shows the learned model of the "guitar-like" quality, over 2 of the the 4 control dimensions. The GP characterizes this quality as peaking when the attack time on the amplitude envelope is short, the "cut-off" on the bandpass filter is low, the frequency of a low-frequency oscillator (LFO) is medium, and the detune on the chorus is high. Colloquially, a guitar-like has a "sharp attack", with some warble in the release, and is coloured with some "detuned harmony". In the space of these four control-parameters, this function is unimodal, which means all paths will be maximized at the same point, as shown in figure 2.3, left.

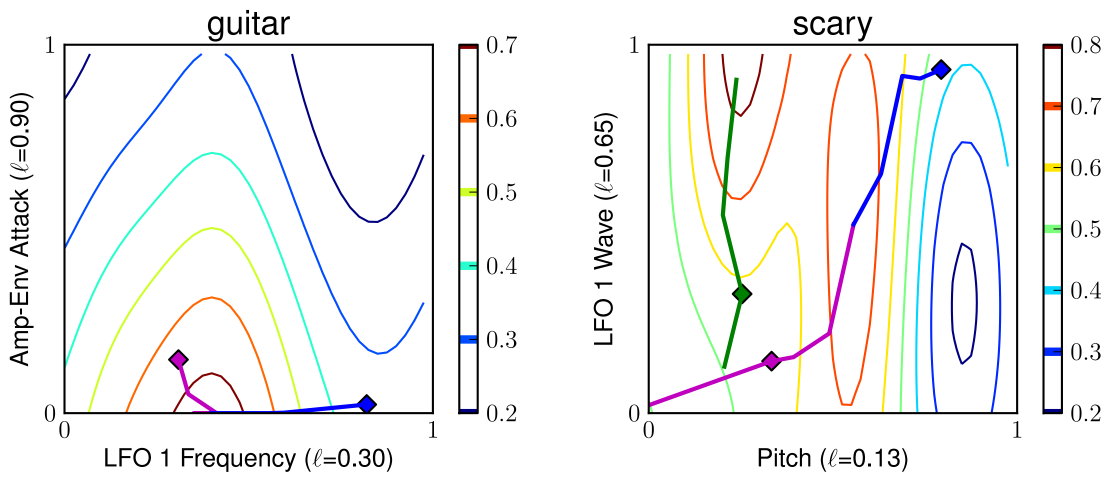


Figure 2.3: Visualizations of high-level concepts. Contours represent the full-Bayes GP posterior mean of the degree of the high-level quality as a function of synthesizer controls. Red, blue and green lines represent paths through control-space which vary the high-level quality, starting from sounds denoted by black diamonds. Left: 2D slice of posterior mean of "guitar-like" function, and knob paths. Right: 2D slice of posterior of "scary" function.

Music or sound concept	GP	SVR	DTR	LINEAR
Pulsating ($n = 64$, $d = 2$)	**0.036**	0.051	0.051	0.073
Guitar-like ($n = 31$, $d = 4$)	0.042	0.029	**0.026**	0.040
Scary ($n = 64$, $d = 4$)	**0.065**	0.068	0.116	0.071

Table 2.1: Comparing the 10-fold cross-validation mean-squared error of different models on user ratings of different high-level qualities. GP corresponds to Gaussian Process, SVR to support vector regression, DTR to decision tree regression, LINEAR to linear regression. n is the number of ratings (values range between 0 and 1 continuously), and d is the number of synthesizer control parameters being varied to create the sound, hence corresponds to the dimensionality of the regression. Each musical (or sound) concept was rated by one user. Users rate sounds by placing sounds on a one-dimensional interface to indicate how much they think a sounds bears that quality. (see Figure 2.2 for interface and Section 2.6 for more details).

Example concept 3: "scary"

We model this quality as a function of four different control-parameters: the depth of a filter envelope, the waveshape and the frequency of the LFO, and the main pitch of the synth. In figure 2.3, right, we show a slice of the mean of a GP conditioned on user ratings. In this slice, both the LFO frequency and the depth of the filter envelope are fixed at 0.7 which corresponds to a steady pulsating rumbling base sound. As the LFO parameter increases, the waveshape becomes more complex, for example going from sine waves, to sawtooth, and finally to a much noisier wave, introducing hisses into the foreground making the sound more scary. As the main pitch of the synth increases, the volume of the "scary" sound body also increases. However, if the pitch is increased too much, the sound becomes much thinner and loses its force. In between the two modes, the foreground hissing and the background rumbling merge, and the sound becomes slightly less scary.

Predicting user ratings

We compare predictive performance of our GP-based model to other models on held-out user ratings. The results are shown in Table 2.1, where we see that for multimodal qualities "pulsating" and "scary", our model predicts user ratings better than support vector regression (SVR) with a radial basis kernel, decision tree regression (DTR), and linear regression. Although the predictive performance of the GP is not uniformly better than other regression techniques, the probabilistic nature of the GP enables active learning, and the smooth function estimates it provides enable us to compute continuous adjustment paths. Future work will revisit the covariance function used in our model, in order to allow it to capture more structure.

2.7.3 Evaluating active learning

To evaluate our path-informed model-based active-learning algorithm, we run simulations on two functions to compare their performances to several baselines. We used IPython's parallel framework to evaluate the expected utility of each candidate in parallel, corresponding to the outer for loop in Algorithm 1. We first started with a simple synthetic function, and then ran another set of simulations on a function learned from user ratings. We show that our method in the former case is able to learn target concepts faster according the metric we define below in Eqn. (2.4) by focusing on regions most relevant to the knob paths. For the latter function, even though our method did not learn faster than some baselines according to our current metric, it performed qualitatively better by being able to identify multiple modes, and to route different starting points to their nearby peaks. In future work, we need to define an error metric that captures the multimodal nature of synthesizer concepts. We also wish to devise a metric that depends on relative perceptions. For example, a user may be less concerned about a knob giving a sound that is exactly 0.5 "scary", if there even exists such a notion, but instead she might be more concerned if moving a knob in one part of its range changes the sound a lot more than some other parts.

As a start, we define *error* as the sum of absolute differences between the desired levels $\mathbf{y}_d$ and the actual levels of the points returned when optimizing for the desired levels, summing over the knob paths for all starting points $\mathbf{X}_s$. $\mathbf{X_i}$ and $\mathbf{y_i}$ denotes the accumulated points the models have evaluated up to iteration i plus all the starting points$\mathbf{X}_s$. f here is the function that simulates the human rating. In the first experiment, this function is a synthetic function, while in the second experiment, this function is the mean of aGP conditioned on the 8x8 grid of user-ratings of the "pulsation" concept.

$$err(\mathbf{X_i}, \mathbf{y_i}, \mathbf{X_s}, \mathbf{y_d}) = \sum_{\mathbf{s=1}}^{\mathbf{S}} \sum_{\mathbf{d=1}}^{\mathbf{L}} |\mathbf{f}(\mathbf{x_{sd}}) - \mu(\mathbf{y_{sd}}|\mathbf{x_{sd}}, \mathbf{X_i}, \mathbf{y_i})| \tag{2.4}$$

We compare our method to several baselines, including proposing points according to the latin hypercubes, and an "entropy" baseline that proposes as the next point the mode on the variance of the posterior predictive marginal of the GP, by running local optimizers from the initial presets. We also compare between several variations of our path-informed active-learning method. "Active learning (re-opt)" re-optimizes the hyperparameters after each fantasized outcome, while "active learning" does not. The "path random" variation skips the fantasizing step, and choose at random a point from the set of proposed candidate points, which were originally randomly sampled from points nearby those visited by the truncated-Newton optimizer while determining the knobs paths. The “path entropy” variation also skips the fantasizing step, and chooses the point that has the highest variance among the proposed candidate points.

In both of the experiments, the number of candidates proposed was 15 and the number of Monte Carlo samples taken for each candidate was 30. The simulations were initialized with simulated ratings of 2 to 3 preset control-parameter settings, which corresponds to the sounds to be adjusted by the knob paths. The experiments were terminated when the performance started to plateau Figure 2.4 or when the model began to pick up the multimodality of the rating function Figure 2.4. For all our active-learning variations, the means and standard errors are averaged across five runs.

Experiment 1: Synthetic function

The synthetic function simulates a concept where most of the space does not give rise to its quality. The function was originally two-dimensional. In order to experiment with how the algorithm would perform in higher dimensions, we artificially added two more dummy dimensions, whose values do not affect the function. Figure 2.4 shows the performance of our method and other baselines as the models iteratively adds more data observations.

Figure 2.4 shows that our active-learning methods are able to learn high-level knobs faster under our error metric Eqn. (2.4). Furthermore, the downward curves of the active-learning methods show that they are able to iteratively improve and refine the model, while for other techniques the lines rise and fall with much higher error. Figure 2.5 shows the different response surfaces at the ninth iteration. We can see that our active learning approach was able to focus the evaluations in the region that is most relevant to the knob paths.

The objective of our active-learning approach is to reduce the uncertainty on the knob paths, as opposed to learning the function everywhere. Figure 2.6 shows how well our "active-learning (re-opt)" method and the "entropy" baseline predicts the true function over the entire space. We evaluate the two corresponding GPs at 1000 locations given by the Sobol sequence, and plot these predicted values against those of the true function. This true function has a lot of low regions around zero, corresponding to the whitespace in Figure 2.5 without dotted contours. Only about a quarter of the space is occupied by a Gaussian bump. This shape is typical of mappings on synthesizers where most of the sounds in a space have none of a certain quality, and only a small part of the space gives rise to that quality. However, a model would have a low mean-squared error if it simply predicted low values everywhere. This is undesirable for constructing knobs, because the knob would be very flat and would

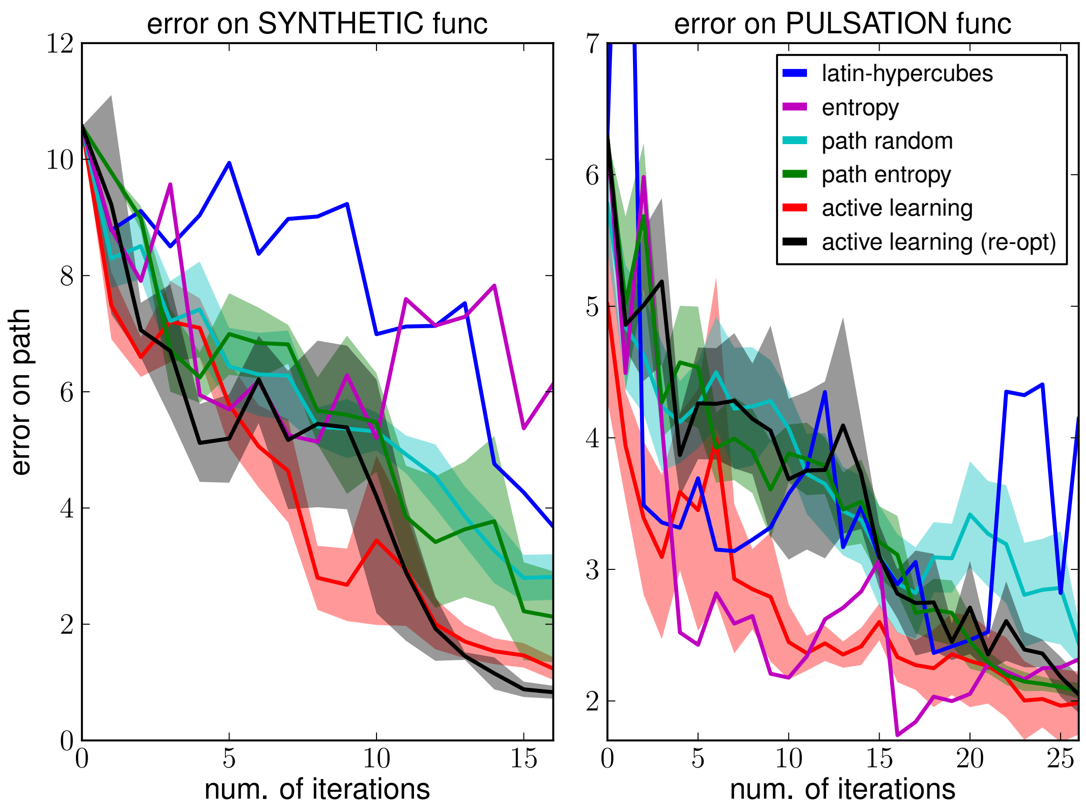


Figure 2.4: *Error* for active-learning variations and baselines on predicting knob paths on synthetic function (left), and "pulsation" function (right). Solid lines represent the mean, and shaded colors show the standard error over 5 runs.

not be able to make much adjustments to sounds. We see that in this case, our active-learning method is still able to predict well over a wider range of levels. Hence, if a user is able to find a sound that begins to have some of her desired qualities, then our tool will be able to give her knobs that can help her increase or decrease that degree, and this adjustment is often challenging for users to perform manually because low-level controls interact.

Experiment 2: "Pulsation" function

The function learned from perceptual data is based on user ratings of sounds in terms of their "pulsation" - a high-level concept where amplitude and frequency modulation interact to give a non-trivial response surface. The user was queried at an 8x8 grid on two synth control parameters, and the mean of a GP conditioned on these ratings was used during comparison of learning procedures. We also artificially added two more dummy dimensions to this function, whose values do not affect the function.

Figure 2.4 shows the performance of our method and other baselines as the models

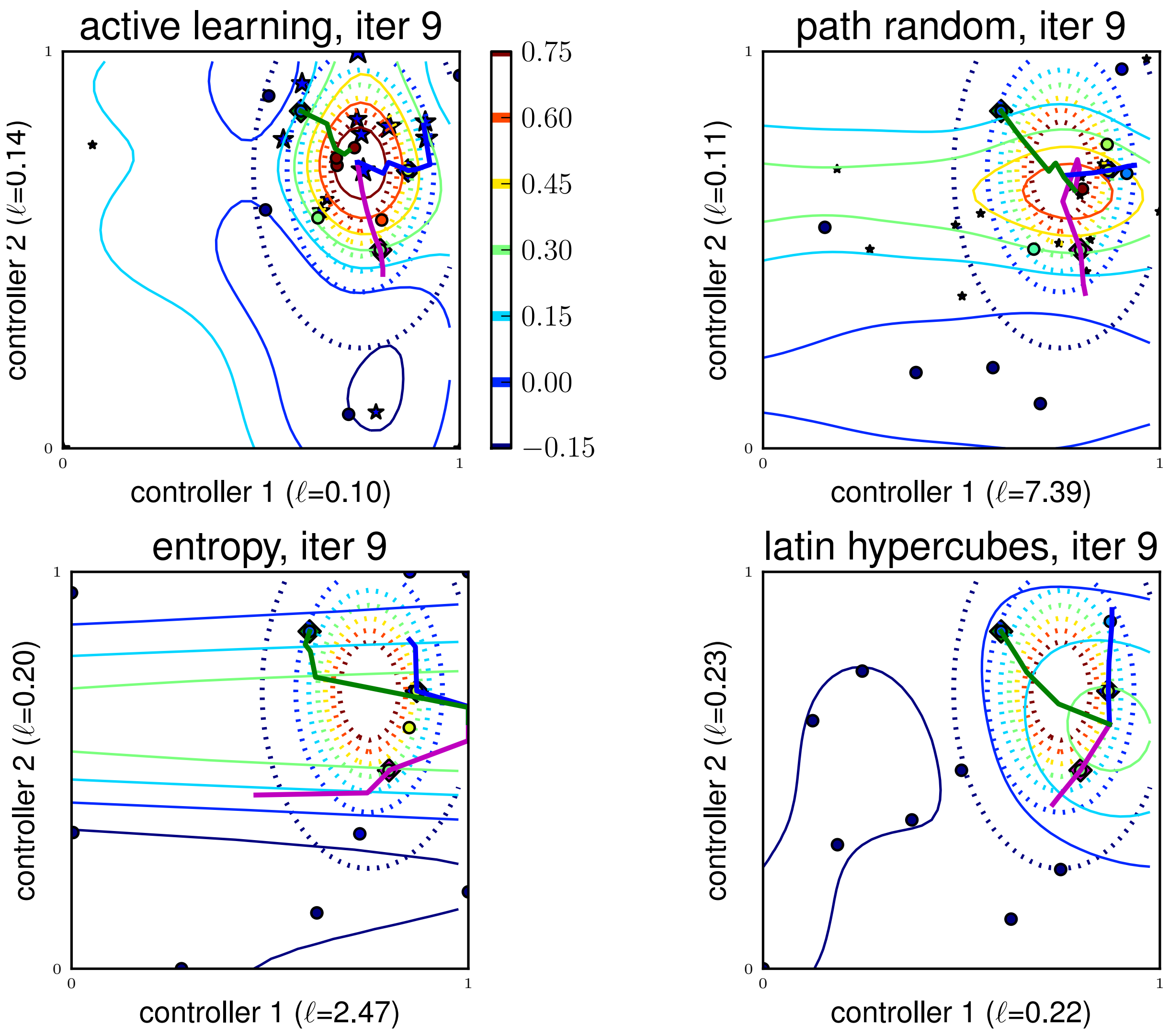


Figure 2.5: In the synthetic experiment, the full-Bayes GP posterior mean for different methods after 9 more observations beyond initialization. ℓs on the X and Y axes give the lengthscales of the corresponding low-level controllers under MLE. Dotted contour lines represent the target function. Dots indicate observations, and stars correspond to proposed candidates, which only applies to the active-learning methods. ℓ is the learned lengthscale of the squared-exponential kernel.

iteratively adds more observations. According to our current error metric, the "entropy" baseline reaches the lowest error the quickest, at iteration 16. However, it was not able to discover that the concept is actually multimodal, as shown in Figure 2.7. In contrast, the best iteration from our active learning (with hyperparameter re-optimization) was able to learn that there was more than one mode in the concept, and routed the starting points to their nearest peak, as shown in Figure 2.7. Figure 2.7 shows the functions learned by the

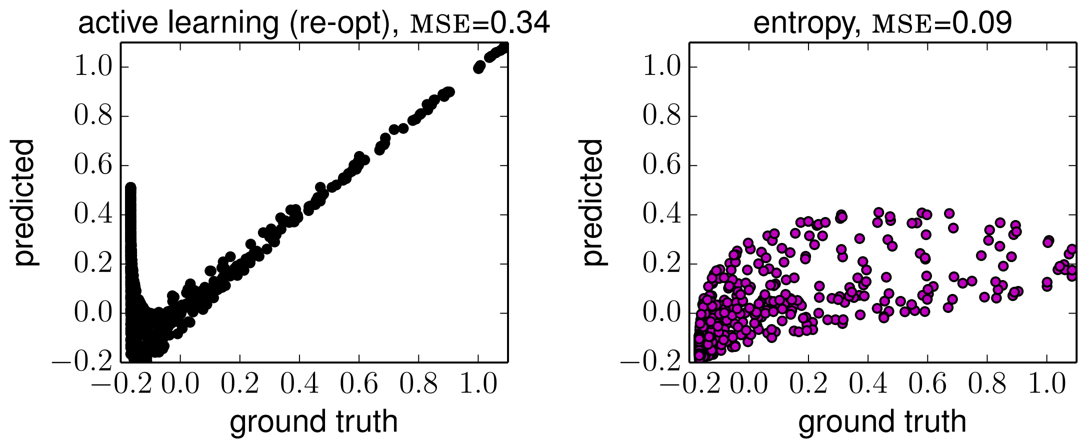


Figure 2.6: In the synthetic experiment, a comparison of how active learning and the “entropy" baseline predict the target function over the entire space. MSE corresponds to the mean-squared error.

different techniques when we terminated the experiments at iteration 27.

2.8 Discussion

We now discuss a few observations and lessons learned from our experiments. Currently, we assume users are able to identify a small subset of relevant low-level controls, and our method focuses on learning the nonlinear interaction between these controls. To scale to higher-dimensional settings, we will have to encode domain knowledge into the covariance structure of our priors. For example on equalizers, adjusting gains for neighboring frequency bins vary perceptually in similar ways, we can parameterize the distance metric in our kernel by the Malahanobis distance to explicitly model covariance between dimensions. Alternatively, if we know the kinds of interaction between controls vary in different parts of the control space, we can call for a non-stationary kernel. See Duvenaud et al. (2013) for insights on how to choose the structural form of the kernel.

Moreover, our method for constructing knob paths is still naive, as it only optimizes separately for different knob values and there is no guarantee that the resultant path is coherent. For example, a path could jump around in the control space due to the many-to-one mapping from control space to musical concept. In the future, we plan to regularize our paths and optimize for a path as a whole so that we can control its perceived smoothness.

Furthermore, our user studies are still preliminary. More human-in-the-loop experiments are needed to better understand how users use our procedure to help them build personal control knobs. Casual observations show that users construct preferences on-the-fly, and they

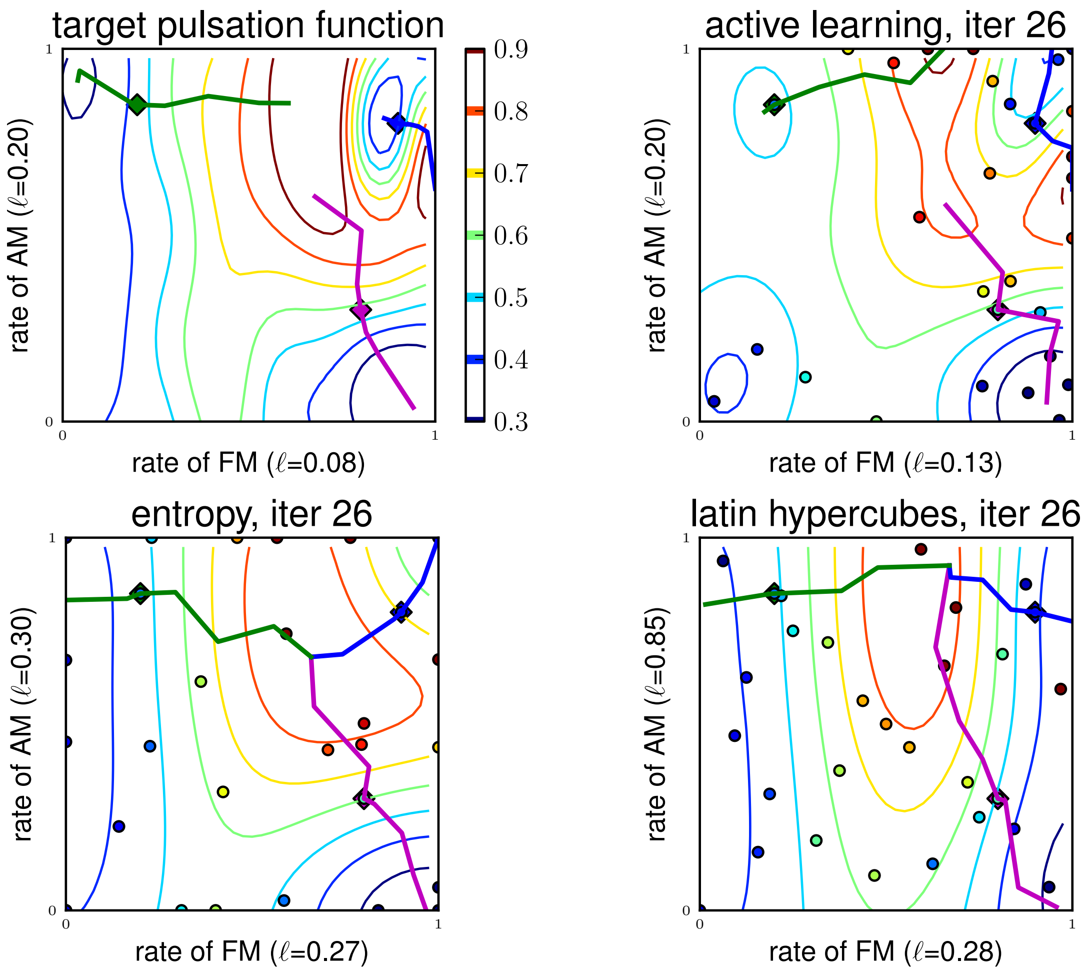


Figure 2.7: In the "pulsation" experiment, the full-Bayes GP posterior means for different methods after 26 more observations beyond initialization. Note the upper-left subplot shows the target function. ℓ is the learned lengthscale of the squared-exponential kernel.

alternate between phases of exploration and refinement, analogous to phases of divergent and convergent thinking (Cropley, 2006). For example, one composer requested sounds from our lightweight "path random" active-learning procedure to refine his "guitar-like" knob, after a dozen of sounds he felt the need to explore whether different kinds of "guitar-like" qualities existed. So he switched to using our entropy-based procedure to try to touch the boundaries of the space, and was pleased that it gave him sounds that he had never heard before. Second, the listening mode of the user often evolves. Initially, users tend to react intuitively to sounds, but later on, they begin to listen more analytically, and explicitly reason and weigh sonic attributes when rating sounds. For example, a sound may have a "sharp attack" like a guitar but may "warble" too quickly to be one, and a composer may still rate it above average because it possess at least one of the main characters of a guitar sound. Third, as user preferences may change and in light of new possibilities, the re-rating of existing sounds becomes necessary, and one composer requested for a feature to "shave off" similar sounds, so that the new ratings and new sounds can have more influence.

2.9 Conclusion

Composers seek to explore sounds along intuitive and personal sonic dimensions, but synthesizers can often only be controlled through low-level controls that interact in complex ways. Inspired by this mismatch, we proposed a novel formulation of high-level knobs that treats a knob as a dynamic mapping that allows us to adjust different sounds along different paths in the control space. In this paper, we presented two building blocks towards realizing such knobs. We adopted the expressiveness of a fully-Bayesian nonparametric model, Gaussian processes, to model the rich perceptual world of synthesizers, where there are many ways to achieve a certain musical quality.

Second, to assist users in finding the set of examples to demonstrate a high-level concept, we derived a model-based active learning algorithm that queries the user in order to improve knob paths, and we showed in simulation that it is more effective for learning high-level knobs. Our procedure is modular, allowing us in the future to swap in different knob path formulations. As there are often multiple ways of achieving a certain desired effect, as shown by the multimodality of our example concepts, we are considering a variation that we call "branching" knobs, which would provide the user with multiple paths for adjusting a sound, some with endpoints at modes further away or higher than others. This allows composers to explore a wider palette of related sounds, and to trade off between achieving a desired adjustment and preserving different aspects of the original sound. Last by not least, our formulation is general, and can be applied to any other domain for users to construct and calibrate a layer of richer, personalized controls on top of the parameters provided by the original system.

Chapter 3

ChordRipple: Recommending Chords to Help Novice Composers Go Beyond the Ordinary

Novice composers often find it difficult to go beyond common chord progressions. To make it easier for composers to experiment with radical chord choices, we built a creativity support tool, ChordRipple, which makes chord recommendations that aim to be both diverse and appropriate to the current context. Composers can use it to help select the next chord, or to replace sequences of chords in an internally consistent manner.

To make such recommendations, we adapt a neural network model from natural language processing known as word2vec to the music domain, as chord2vec. This model learns chord embeddings from a corpus of chord sequences, placing chords nearby when they are used in similar contexts. The learned embeddings support creative substitutions between chords, and also exhibit topological properties that correspond to musical structure. For example, the major and minor chords are both arranged in the latent space in shapes corresponding to the circle-of-fifths.

Our structured observations with fourteen music students show that the tool helped them explore a wider palette of chords, and to make "big jumps in just a few chords". It gave them "new ideas of ways to move forward in the piece", not just on a chord-to-chord level but also between phrases. Our controlled studies with nine more music students show that more adventurous chords are adopted when composing with ChordRipple.

■ 3.1 Introduction

Novice composers often find themselves consciously or unconsciously falling back to familiar sets of chords, because these chords are what they are "used to hearing" and where their

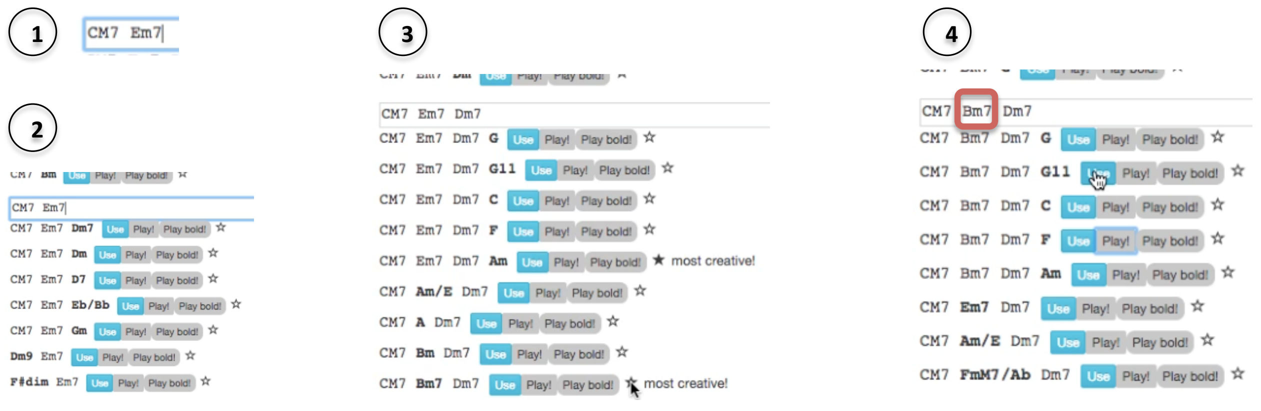


Figure 3.1: A walkthrough, starting with two chords (step 1), of how ChordRipple is used to help inspire the next chord (steps 2, 4) and for exploring substitutions (step 3, where the bottom four choices are substitutions of the second chord, which are from the Ripple effect of choosing the third chord. As shown, chord recommendations here include both single substitutions and also chords that change the context (what is around a chord change), namely Ripples.

fingers are used to falling on their instrument. In the corpus of 200 rock songs curated from the *Rolling Stone Top 500 Hits* Temperley and Clercq (2013), we see a long-tail phenomenon where a few chord progressions are used in many songs while most other chord progressions are rarely used, akin to the Zipf distribution of words in natural languages. In fact, a recent article showed that the evolution of our harmonic language, among other musical facets in contemporary western popular music, has stagnated for more than fifty years Serrà et al. (2012).

However, the challenge is often not only to find something that is novel, but to use these chords in a way that serves a personal musical impetus. In practice, novice composers often find it difficult to use more elaborate chords because they are less certain about their effects and how to integrate them into their own music in a way that feels both original and connected.

To address this challenge, we are developing ChordRipple (shown in Figure 3.1 and short-handed as CR)—a system that helps composers be more adventurous by recommending chords that are similar to the chords that they are using thus preserving the composer's original musical intent, but that are a little more unusual. By proposing such chords as possible alternatives to the composer's current chord choice, the system helps the composer see less familiar chords in the light of their own musical discourse, allowing them to easily experiment with the alternatives by substituting them into the current musical context.

We introduce a novel intervention called Ripple, which are chord recommendations that not only suggests alternative chords to the current chord but also recommends a reworking of its content. The name Ripple is an conceptual analogy to rippling effects, which in this case is the changing of one chord causing the change of its surrounding chords.

Chord recommendations are enabled by a machine learning model that captures chord similarity based on how likely different chords are to be used in similar musical contexts. We trained this model on a corpus of 200 rock songs curated from the *Rolling Stone Top 500 Hits* Temperley and Clercq (2013). This model allows the system to recommend chords that are novel for a particular musical context but still similar to the chords used by the composer.

To evaluate how CR impacts composers' creative process, we recruited 14 undergraduate music students, and asked them to use CR to further develop chord progressions for a piece they were working on. We observed both positive and negative impacts of the tool on the creative process. Positive impacts include the tool helping composers explore a wider palette of chords, to create chord progressions that were "fresh and different throughout". It gave them "new ideas of ways to move forward in the piece", not just on a chord-to-chord level but between several phrases. It encouraged them to make "big jumps in just a few chords". They sometimes altered their intentions of extending or closing a phrase based on the recommendations they saw. Instead of following the path of least resistance, the recommendations often made them think and work harder. However, some composers found the choices to be too many and that not all chord recommendations were always relevant to what they were trying to achieve. We propose that future systems of this kind should focus recommendations to what is most relevant under the composer's current intentions. For example, the intention of "make it longer" versus "I'm trying to end here" would entail very different chord recommendations.

As the first study was more exploratory and open-ended, we conducted a second study that was more controlled where students performed transformations on given chord sequences with three design variations of ChordRipple. We found that the presence of adventurous chord recommendations resulted in students composing more novel chord progressions. Yet we did not observe that recommending more difficult to use chords with Ripplesallowed students to adopt more of them when compared to just recommending the chords themselves.

The contributions of this paper are threefold. First, we built CR, a creativity support tool that helps composers try more adventurous options; CR supports both adding new material to an existing chord sequence and re-editing existing material (thanks to Ripples). Second, we adapted a machine-learning model to learn chord similarity based on actual chord usage. Third, we conducted evaluations that yielded insights into how tools like CR can be integrated into composers' creative practice.

■ 3.2 Related work

There have been a number of creativity support tools that assist novice composers in composing chord progressions. Most systems are designed to address the musical problem of harmonizing a given a melody Chuan and Chew (2007); Simon et al. (2008); Nichols et al. (2009); Morris et al. (2008).

While these systems all use a data-driven approach, each system strives for a different goal and objective in recommending chords. For instance, Simon et al. (2008); Morris et al. (2008) allows users to adjust the overall "mood" quality of the accompaniment through knobs such as "happy" which adjusts the weight between the transition matrices of a hidden Markov model for the major and minor key. Users can also modify chords individually, but the recommended chords are restricted to only the most typical chords given the local musical context. Nichols et al. (2009) provided users with a novel interface with control axes that corresponded to principal components of variances of differential transition dynamics in western popular music, allowing users to more effectively explore a wide range of variations on accompaniment sequences.

Chuan and Chew (2007) takes a hybrid approach of also using music-theory to guide the choice of chords for accompaniment. Fukayama et al. (2013) works with chord sequences by themselves and by performing non-negative matrix factorization on a windowed transition matrix, the system allows users to interpolate between different chord progressions to create their own mix.

CR instead presents a composer with sets of diverse chords that are appropriate but more rarely used in a particular musical context. The support for exploring novel and diverse inspirational examples is critical because they help users to generate more novel and diverse artifacts Marsh et al. (1996); Nijstad et al. (2002); Siangliulue et al. (2015).

■ 3.3 Chord2Vec

CR provides two mechanisms to help composers adopt novel chord choices in their compositions. First, CR recommends chord substitutions ranging from typical to adventurous (yet still musically appropriate). Second, because the change of one chord in the middle of a phrase may cause the need to change its context, (a concept we name the Ripple effect), the system also includes chord recommendations of varying lengths that change a chord's surrounding chords. We designed the first mechanism to help composers create raw adventurous material, while the second mechanism to make it easier to for composers to make adventurous revisions to existing material.

To build a system capable of recommending chords, we first need a model of chord transitions and similarities. To represent chords, we use chord symbols that abstract chords into attributes, such as its root, chord type (such as major, minor, diminished), inversion and bass, extensions and alterations, which can amount to many possible unique chords.

Like words in natural language, the distribution of chords follows a Zipf-like distribution. For example, Figure 3.2 shows chord frequency in the Bach chorale corpus (BACH) in *music21*. A handful of chords are used very frequently, while many chords are rarely used.

Modelling rarely-used chords and chord transitions is difficult. For some applications it is common to reduce the number of chords to 24, treating all chords as either major or minor rooted in one of the 12 pitch classes. However, this comes with the compromise of conflating chords that bear very different transition dynamics. For example, even though `I64` is an inversion of `I`, it serves a very different function in a cadence. Moreover, as our goal is to make adventurous chord recommendations, we need to be able to model how rare inversions and extensions are used. Hence, we keep all the chord attributes as annotated, which for example results in 92 unique chord symbols in the BACH corpus.

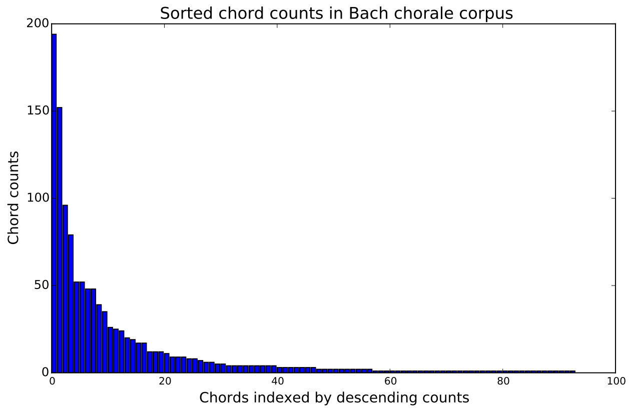


Figure 3.2: The occurrence count distribution of chords in roman numerals used in Bach chorales as annotated by Tymoczko Tymoczko (2010) in the *music21* Bach-chorale corpus Cuthbert and Ariza (2010).

3.3.1 Model

To model chord transitions and similarities, we adapt a neural-network model from natural language processing known as Word2Vec Mikolov et al. (2013) to the music domain as Chord2Vec. This model learns chord embeddings from a corpus of chord sequences, placing chords that are used in similar contexts nearby in a vector space. The learned embeddings support substitutions between semantically similar chords. This allows chords that are rarely or never seen in a particular musical context to be recommended if they are nearby in the embedding to other chords that are commonly used in that musical context.

As shown in Figure 3.3, this model is a skip-gram neural network. Hyperparameter m specifies the dimensionality of the continuous latent embedding, and w specifies the size of the considered context. Latent embeddings of each chord are defined by matrices U, indexed by the input chord, and V, which is indexed by $t+l$, where $-w \leq l \leq w$ indicates output chord c_{t+l}'s relative time position in a sequence with respect to the input chord c_t.

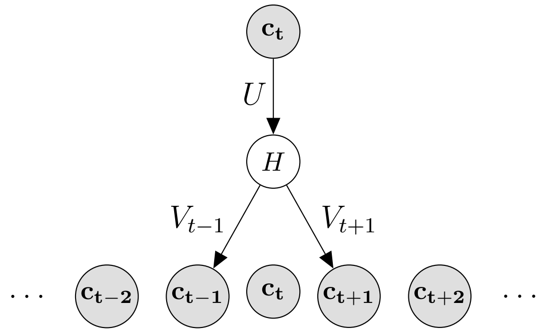


Figure 3.3: Skipgram neural network with window size $w = 1$.

We use u_i to denote the vector representation of chord i. We use c_t to refer to the chord used at time t, and u_{c_t} and v_{c_t} to refer to the input embedding (U) and output embedding (V) of the chord c_t. Following Mikolov et al. (2013), we define $p(c_{t+l}|c_t)$ as the softmax of the dot product between a chord and its context, as in Eqn. (3.1), where $-w \leq l \leq w$. The training objective of the model is to maximize the log-likelihood of all chord sequences independently, as in Eqn. (3.2). Maximizing this objective has the side-effect of puttings chords used in similar contexts close to each other in the input embedding U.

$$P(c_{t+l}|c_t) = \frac{\exp\left(v_{c_{t+l}}^T u_{c_t}\right)}{\sum_{i=1}^{|W|} \exp\left(v_{c_i}^T u_{c_t}\right)} \tag{3.1}$$

$$L(c_0 \dots c_{T-1}) = \sum_{t=1}^{T} \sum_{-w \le l < w,\ l \ne 0} \log P(c_{t+l}|c_t) \tag{3.2}$$

Training data and exploratory experiments

To support ChordRipple, we trained Chord2Vec on a corpus of 200 Rock songs (all transposed to `C`) from the *Rolling Stone Top 500 Hits* Temperley and Clercq (2013) containing 98 distinct chords, using latent dimension $m = 10$ and window size $w = 1$.

In our experiments, we use the two aforementioned annotated corpora, *Bach Chorales* from *music21* Cuthbert and Ariza (2010) annotated by Tymoczko Tymoczko (2010) and *The Rolling Stone: Top 200* corpus transcribed and annotated by Temperley and Clercq Temperley and Clercq (2013). To model the rich palette of chord alterations, we retain all chord attributes from the original chord annotations. We experiment with both transposing chords to the key of `C` versus preserving the chords in their original keys. By transposing we assume that key does not have a major influence on how chords are used. By keeping songs in their original key one can also model key-specific trends at the expense of having less counts on each chord.

Visualizing the axes of the latent embedding

To gain more intuition on how the latent embeddings capture musical semantics, we trained a toy bigram version of Chord2Vec with hidden layer size $m = 1$ for ease of interpretation. We reduced the number of chord symbols by requiring that they have at least 5 occurrences, which left 30 chords. We can see that these 1-D embeddings approximately capture functional harmony. In Figure 3.4, we see that chords close in U share similar target chords, for example, `V` and `viio` chords and its inversions and seventh chords are closely related, as they are often followed by some form of tonic chords. In Figure 3.5, chords close in V serve as similar target chords, for example as a phrase moves into a cadence, chords that transition to a `I64` are likely to choose `ii7` as an alternative or directly move to a `V` chord.

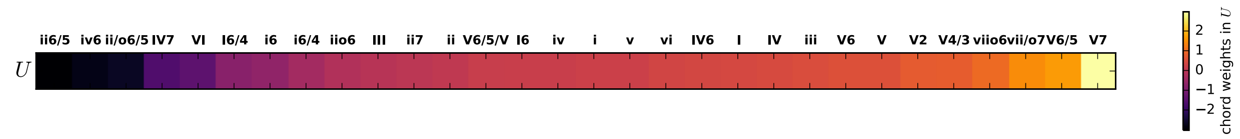


Figure 3.4: The inputs weights U of the bigram Chord2Vec on the Bach chorales, with latent embedding size $m = 1$. Some predominant and subdominant chords cluster on the left while all the dominant chords cluster on the right.

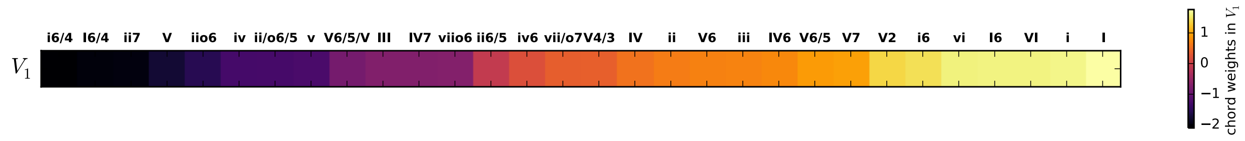


Figure 3.5: The output weights V of the bigram Chord2Vec on the Bach chorales, with latent embedding size $m = 1$. Tonic-like chords cluster on the left end while mostly predominant and subdominant chords on the right

3.3.2 Circle-of-fifths emerge from trained embedding

When training Chord2Vec on the Rock corpus with songs kept in their original keys, the learned embedding exhibits topological properties that correspond to musical relationships. The model had a hidden layer size of $m = 10$ and window size of $w = 1$. The x and y axes in Figure 3.6 correspond respectively to the first two principal components of the projection layer U. It shows that the major and minor chords are both arranged in the latent space in shapes corresponding to the circle-of-fifths. One interpretation of this result is that the tonic and dominant tension relationships are relatively strong in songs of all keys, causing them to push each other apart, some stronger than others causing the angles of the polygon to be different.

3.4 ChordRipple

CR recommends a mix of adventurous and commonly-used chords based on the current context. To recommend more adventurous chords, we use the latent embedding learned from Chord2Vec to query chords that are the most similar to a composer's current chord choice.

Following Mikolov et al. (2013), we define the similarity between chords through the cosine distance of their embeddings vectors, as in Eqn. (3.3). Hence, given a chord c_i, we can query the embedding for the k closest chords to c_i to recommend to users as substitutions, by starting with Eqn. (3.4).

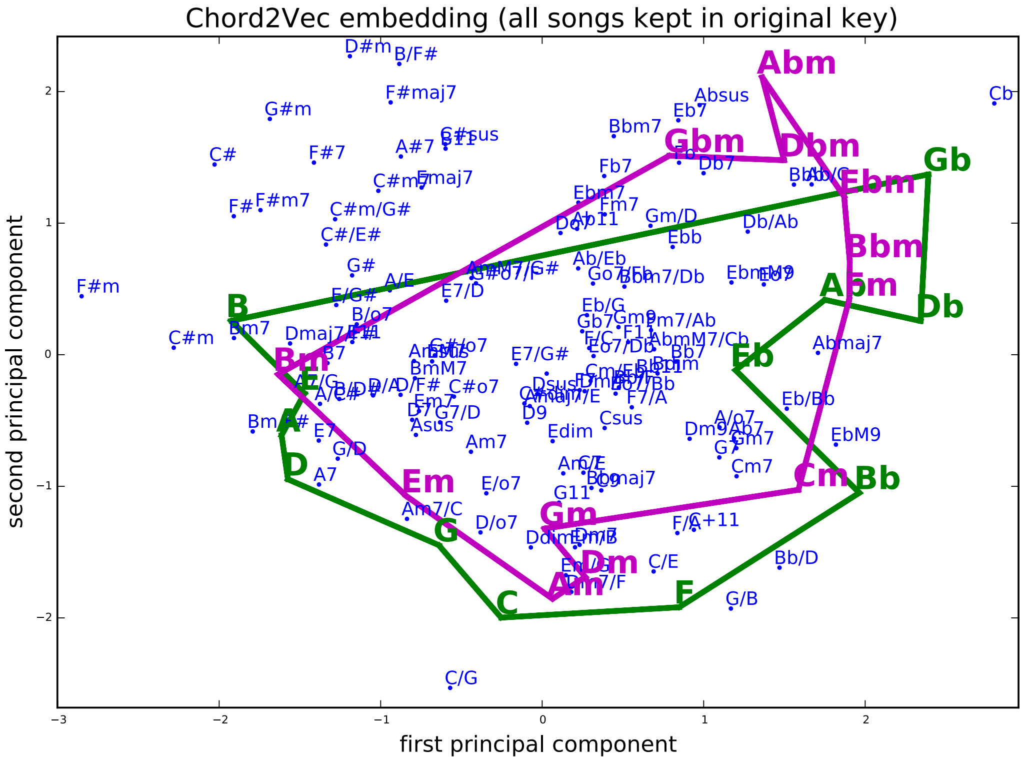


Figure 3.6: The CHORD2VEC projection layer U with hidden layer size $m = 10$ projected onto its first two principal components. The model was trained on the ROCK corpus with songs kept in original key, and then. Major and minor chords are both arranged in the latent space in shapes corresponding to the circle-of-fifths.

$$\text{similarity}(c_i, c_j) = \frac{u_{c_j}^T u_{c_i}}{||u_{c_j}||\ ||u_{c_i}||} \tag{3.3}$$

$$\text{mostSimilar}(c_i) = \arg\max_{c_j} \text{similarity}(c_i, c_j) \tag{3.4}$$

These queried chords are not necessarily the most typical in the current musical context as preference has been put on being similar to a composer's current chord choice. To recommend what is typical given a musical context, we learn a distribution $P(c_t|c_{t-1})$ using a simple bigram model. WORD2VEC also learns a similar probabilistic model, but conditioned on both c_{t-1} and c_{t+1}. For simplicity in the current implementation we adopt a bigram model for these queries.

3.4.1 Ripples: multi-change recommendations

We call recommendations that affect more than one chord RIPPLES. A RIPPLE can help smooth out transitions between chords, making it potentially easier for users to swap in more adventurous chords. It can also give new ideas of how to move forward or backward from a chord. The basic RIPPLE recommends both a substitution for the current chord (c_t) and also a reworking of its context. There are two steps involved in generating such RIPPLES. First, the substitution (c_t^s) is generated either by querying the CHORD2VEC embedding or by inward conditioning on the current context, in this case (c_{t-1}, c_{t+1}). Next, the immediate current context (c_{t-1}, c_{t+1}) can be reworked by conditioning on its larger outer context and the newly substituted chord, (c_{t-2}, c_t^s, c_{t+2}).

We can derive different kinds of ripples through a cascade of inward and outward conditionals. For example, one can imagine regenerating the larger context $\{c_{t-2}, c_{t+2}\}$ by conditioning outward from the newly substituted context (c_{t-1}^s, c_{t+1}^s). For each of the conditioning and substitution steps, we can also imagine sampling them to generate more variations.

We initially ranked the chord recommendations by the transition likelihood (Eqn. (3.2)) of the new sequence, using dynamic programming under the bigram model. Table 3.1 gives an example of chord recommendations from CR when given the most basic chord progression, `C F G C`. We clearly see that the recommendations from a purely NGRAM model are much more conservative, consisting of mostly `C` and `F` chords, while WORD2VEC is much more adventurous, recommending chords that can help composers break out of commonly used chords.

C F **G** C	single sub, *above*	triplet subs, *above*	outward subs, *below*
N-GRAM	**C**, **F**	C **F C F**, C **C F C**	C **C** G **F**, C **Am** G **Am**
WORD2VEC	**Ab7**, **Dm7**	C **F Ab7 Cm**, C **C Dm G**	C **F7** G **B**, C **F/A** G **C7**

Table 3.1: Comparison between chord recommendations from N-GRAM and WORD2VEC the G chord, third in the chord sequence C F **G** C. The chords in bold function as substitutions (sub) for the original sequence. The labels *above* and *below* indicate where these recommendations would appear relative to the input textbox in CR.

However, if we only consider how well the ripple recommendations flow in and out of the substitution chord, we will be neglecting the original structure of the sequence. For example, in Table 3.2, when regenerating the context for the chord F in the sequence C F Dm G C, the smoothest path to transition into its substitutions F or F/A is to precede and follow it with the chord C. The result is not very exciting and by returning to C it breaks down the phrase into two sub-phrases.

To preserve the original structure or add meaning variation to that structure, we need to also consider the similarity of the chords in the regenerated context to that of the original. This can be achieved by maximizing both the transitional likelihood between the substitution chord, it's regenerated immediate context and the larger original context and also the distance between the immediate context to the substitution chord in the regenerated sequence and the original sequence, as shown in Eqn. (3.5). We denote the new sequence with substitutions as $\mathbf{C^s}$ and the original sequence as $\mathbf{C}$, and use k_1 and k_2 to denote the start and end index offsets for the sequence of substitution. λ controls the balance between smooth transition and similarity to original sequence. The is the Ripple recommendation implemented in the second version of ChordRipple where we used $\lambda = 1$.

$$\text{cost}(\mathbf{C^s}, \mathbf{C}) = -L(\mathbf{C^s}) + \lambda \cdot \text{sim}(\mathbf{C^s}, \mathbf{C}) \tag{3.5}$$

$$\text{sim}(\mathbf{C^s}, \mathbf{C}) = \sum_{-k_1<l<k_2} \log \text{similarity}(c^s_{t+l}, c_{t+l}) \tag{3.6}$$

The example in Table 3.2 shows the advantages of this additional constraint. Not only does it regenerate the context with chords that respect more the original function of the chord, it also regenerates the context in a way that is stylistically consistent with the substitution chord (using a seventh chord C7 to lead into F7).

Ripple method	Generated sequence
Transition likelihood only	**C F/A C** G C
	C F7 C G C
Similarity to context	**Cm F/A Ab** G C
	C7 F7 D G C
Original sequence	C **F** Dm G C

Table 3.2: Comparison between ripple recommendations that only take transition likelihood into consideration when regenerating the context, versus recommendations that also consider the similarity of the regenerated context to the original

3.4.2 User interface

To make it easier for composers to experiment with these recommendations, we designed a user interface that allows composers to see alternatives in relation to their own choices, akin to autocompletions. As users type their chord sequences into the input text box shown in Figure 3.1, suggested chord sequences appear in parallel both above and below, with chord changes highlighted in bold. Users can listen to how their chord sequence might sound if they had chosen a particular suggested sequence. More typical chord recommendations are near the text box, while more adventurous ones are at the fringes. Composers can use the recommendations to help select the next chord, or they can go back to earlier chords and see the bolded changes ripple backwards.

As shown in Figure 3.1, chord sequences below the text box are recommendations for what can be changed around the current chord. For example, the first five rows below show the top five typical continuations. The bottom two rows show how the chord before the most recent chord be approached differently, which in step 2 of Figure 3.1 is the second chord in the sequence. Cropped from above the text box are pairs of recommendations that include a single substitution for the current chord and triple substitutions for the chord and its contexts. The top half are the adventurous ones and the bottom half the typical. By clicking on the Use button, users can easily replace their own chord sequence with one of the recommended. A walk through of Figure 3.1 is given in the next section.

3.5 User evaluations

We conducted two user studies in order to understand how CR impacts the compositional process, each with different degrees of emphasis on external and internal validity. First, we wanted to understand how music students would relate to the the tool, and in what ways the tool can assist them in composing chord progressions. We also wanted to observe if the tool can help students go beyond their usual palette of chords. In this exploratory study, we asked students to use the tool to help them create variations of chord progressions in their own music. They were hence more intrinsically motivated. For the second study, we conducted a controlled experiment to specifically study the effects of adventurous and Ripple recommendations respectively on the novelty of students' chord progressions and their satisfaction. Each student was asked to perform transformations on given chord sequences with three design variations of CR.

■ 3.5.1 Experiment 1: Structured observations

We want to understand how CR impacts the compositional process. What kind of goals and expectations do composers have when working with the tool? How does the tool support or alter their intentions? How do they choose between different chord recommendations? To answer these questions, we took a structured-observation approach Garcia et al. (2014) to our studies. We recruited 15 participants to use CR to further develop the chord progressions they were working on for a piece. We logged their edit traces. We interviewed the participants both before and after the composing task. For the last ten participants, we audio-recorded and transcribed their interviews, and screen captured their interactions with CR.

Participants

We recruited 14 undergraduate music students and a graduate student in the applied sciences with an interest in music. The total of 15 participants included five females and ten males. We paid them $10 each for a 55-minute study session. Fourteen participants were undergraduate music students from three local universities. All the music students had some composing experience, ranging from composing for harmonization assignments in music theory classes to writing their own symphonic pieces. Most students played either the guitar or the piano, and used the instrument as a means to composing chord sequences. Most students composed roughly in the genres of pop, rock and jazz, or with traditional classical harmonies.

The composition tasks

We asked each participant to bring in a piece that they were currently working on that involved chord progressions. We suggested three possible tasks for using our tool: (1) to create variations of their existing chord sequences, (2) to use a few chords from their chord sequences as a seed to growing a new sequence, (3) to compose a new chord sequence from scratch. These tasks were structured enough to provides us with some degree of uniformity for comparison and generalization across participants, but still open enough to support individual expression.

Procedure

Each participant was asked to bring their scores and sketches of a current piece they were working on, with the chord progressions they used in guitar-tablature-like notation. Each session began with a short semi-structured interview on their background in music, how they compose, the role of chords in their music. They were then asked to walk-through how they composed the piece that they brought in. After the initial questions, participants were introduced to CR. To familiarize them with the chord notation used by the interface, participants were asked to type in the chords sequences that they brought in. We then suggested to them three possible short compositional tasks to perform with CR, which we will further outline below. Depending on how much time a participant took to complete the first task, each participant typically completed one to two of the tasks. During the task, participants were asked to adopt a think-aloud protocol. To further understand why and how participants were making decisions at particular moments, we interjected with probing questions. After each task, we conducted a short semi-structured interview for retrospective verbal accounts of how they felt the tool had impacted their creative process.

Two detailed walkthrough of ChordRipple usage

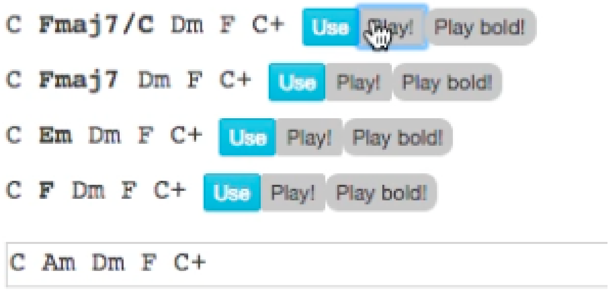
We start by walking through a long and short example to illustrate how composers use the tool, how they make decisions, and the impact of the tool on their creative process and the creative outcome.

Composing a second phrase as a variation

Participant 7 (P7) came in with the following chord progression for a minimalistic piece, where this sequence would be looped: C Am Dm F C+. He wanted to explore what else he could do with it. Before he started, he decided he wanted to keep the sound of the augmented chord C+ at the end to preserve the identity of the phrase, but "break out of the really diatonic stuff in the middle". When seeking substitutions for the second chord, he manually replaced Am with Em, which are both possible tonic substitutions that can function as tonic prolongations of C. In this case the composer explored closely related chords, while the tool suggested a more unusual alternative Bb.

> C **Bb** Dm F C+: "This one is not diatonic, [it makes] an aggressive shift, which I like. I feel like that's something that could be used as a climatic sudden moment not so much as a thing that can be repeated".

Even though he liked the effect of Bb, it was not appropriate for the current context as the chord sequence at stake was supposed to be looped, but the suggested chord was meant to be a one-time "sudden surprise change". He then reverted back to the second chord being Am. Figure 3.7 shows the alternatives recommended by the tool. The chord recommendations Fmaj7/C and Fmaj7 on the top two row are the chords that are most similar to Am according to the WORD2VEC model. In this case, these chords are also mutually similar, being inversions of each other. The bottom two chords, Em and F, are the most likely chords given the current surrounding context being C and Dm. In roman numerals, they would be `I iii ii` and `I IV ii`, the latter being more common than the former. The composer eventually chose Fmaj7/C, which holds the bass note of the previous chord C. He liked that the chord was an inversion and that it was a major 7 chord, saying "It's a good sound. It's different from what I am using".


```
C Fmaj7/C Dm F C+   Use  Play!  Play bold!
C Fmaj7 Dm F C+     Use  Play!  Play bold!
C Em Dm F C+        Use  Play!  Play bold!
C F Dm F C+         Use  Play!  Play bold!

C Am Dm F C+
```

Figure 3.7: Chord recommendations for Am, the second chord in the sequence C Am Dm F C+. The top two rows are SIM type recommendations, while the bottom two rows are IN-RIPPLES.

At the end, the composer came up with two new phrases for his piece, C Fmaj7/C Gm7 F C+ and C Fmaj7/C Bb C Cm G. For the second to the last chord, he chose Cm because he realized it links to two phrases with an interesting internal logic, that is, C "opened" up to go to C+ in the first phrase, while in the second phrase C went in the opposite direction and "contracted" to Cm. The tool gave him "ideas of ways to move forward in the piece, previously I just had this as a repeating progression". He felt the chord sequences were consistent with his style in a way that they bore "grains of progression" that [he] uses a lot, but at the same time the tool helped him go beyond the ordinary. The tool helped him to explore the in-between space:

> "If I am trying to come up with something tonal but [with] variation ... I have difficulty. It's either really tonal or its not and I am not good at the in between, sort of like tonal but breaks free, so that's I think what I accomplished here a little bit".

Using Ripples to change what was fixed

Participant 8 (P8) came in with a 12-chord long sequence. He had come up with the first two chords, `CM7 Em7`, on the guitar and he "felt like [he] could go somewhere with it" and that he could "really make that build". But when he went to the piano, it took "a different direction".

For the study, we decided to start with the same two chords, `CM7 Em7`, and see what other directions the tool would take it to. The composer initially worked forward, exploring continuations (Figure 3.1 STEP 2), and then chose `Dm7` as the third chord. In Figure 3.1 STEP 3, he continued to work forward, and marked the most untypical continuation as the MOST CREATIVE . But upon seeing that there were also recommendations for how to change the existing chords (bottom four rows), he chose `CM7 Bm7 Dm7`, which replaced the original second chord `Em7` with `Bm7` (result shown in Figure 3.1 STEP 4). This recommendation was based on its surrounding chords `CM7` and `Dm7`. Note that `Dm7` was a chord inspired by the tool earlier.

Even though the composer originally thought of the first two chords as fixed, he was flexible and open to changing them when he saw a substitution he liked. This example also illustrates how the tool could inspire a cascade of changes.

Overall expectations and attitudes towards ChordRipple

Participants' reactions to CR were mixed. We give a few brief examples of how participants reacted to CR for the different tasks. For the task of seeking variation to an existing chord sequence, we saw two contrasting intentions from composers. For example, P14 was open to new ideas of how the sequence could be varied: "I was more open to changing up the direction of it or the feeling of it". Meanwhile, for P7, it was the opposite. He had a more specific idea about what the desired variations would sound like and how they would relate back to the original sequence: "[I am] more picky about how I am changing it. [It] took longer [than composing from scratch] to go through [the chord recommendations] and find the right chord".

For the task of starting from scratch, we also saw two contrasting experiences. For example, P7 felt that composing with CR was much faster than improvising on a piano: "I like it when I'm composing from scratch. I have less an idea of what I want, hearing what's next, makes the process quicker". In contrast, P15 found it difficult to generate a chord sequence from scratch, partly because this way of working was foreign to him. He usually writes songs by collaborating with his friend who would first compose a melody and pass it on to him, and he then would come up with chords to accompany it.

Choosing between chord recommendations in ChordRipple

There are often several different levels of decisions at play when a composer is trying to decide which chord to choose. For example, Delalande (2007) outlined four decision levels in a compositional process, which consists of the objective, the musical idea, grammaticality and technical aspects. Moreover, a composer's decision-making criteria are dependent on the stage she is in in the compositional process and also where she is structurally in the piece. Furthermore, a composer might be trying specifically to find the chords she is hearing in her mind [P9] or be open to new ideas [P14].

The different decision levels are often interconnected. A composer may have an overall objective of "breaking out of diatonic harmony" [P7]. Similarly, P14 says "I'm looking more for less conventional chords". These aesthetic aspirations can impact what composers consider as grammatically plausible. On the other hand, a certain chord may work well grammatically but "changes feel" [P14], changes what the phrase is expressing. This elevates the level of decision to that of the musical idea.

When choosing between recommendations in CR, composers often anticipate what subsequent changes or continuations may be needed. For example, choosing one chord may lead to the need to change the next two chords [P14] because it does not flow well into those two chords which are themselves a unit. P1 struggled between a chord she liked more but would require more work to complete the phrase, versus a chord that would conclude the phrase immediately.

Impact of ChordRipple on the composition process

Upon first encountering the tool, many participants felt compelled to listen to all of the recommendations. P14 commented "I want to listen to all of them, even if it might not make sense just looking at it, because why not, maybe it will spark something". P15 felt the options were too many and that it was hard to choose between them. It was "like too many ice-cream flavours when you only want three". In contrast, P9 said "I use this like an inspiration and so seeing, having it suggest a ton of stuff to me, I was just like getting more information faster".

Participants had different interpretations on the nature of the recommendations and their effects. P9 described the tool as "yelling at you with things you won't think about", and when he "sees something crazy, try it, get more crazy". This sentiment was echoed in P6. Upon seeing a chord recommendation that increased tension, she was inspired to want even "more intensity, so [she] added a 7th'. P12 said the tool helped him "see where it can go from here". P9 likened the tool to a co-pilot that gives him validation when he sees the chords he wants to try out among the suggested chords. Many composers were delighted to explore a wider palette of out-of-key, extended, altered and inverted chords. P7 said the tool helped him stay "out of straightforward diatonic space", and that it suggested "new places to go". While others found some of the out-of-key chords to be irrelevant: “It doesn’t go together. I just feel like it’s random” [P14].

Let us zoom into out-of-key chords for a moment. Composers find it harder to workout out-of-key chords manually: “It is hard to generate these chords without hearing how they sound first” [P8]. CR helped composers imagine and hear how to transition to and from and between out-of-key chords, making them more accessible.

> “That’s a big jump for me to make in just a few chords, something I’m not able to do on my own, play around and land there” [P7].

P7 pointed out that CR helped him infer which key he was in by suggesting continuations that showed him “this is where you are going” [P7]. He said he would otherwise have forced the sequence to end in the wrong key and destroyed it: “it saved me from myself”, said P7.

P8 described the experience of working with CR as faster than working things out in his head, while P6 felt it was a much faster process then improvising on a piano:

> “It is just faster, instead of doing that in my head, these are all the options that make sense, just try them and see if one of these are what you are thinking about” [P8].

The tool at times altered the workflow of the composer by inspiring new directions. For example, P7 originally planned to fix the beginning and the end of the phrase and just work on the middle. But upon seeing the chord recommendations for the middle chords, he chose chords that suggested new directions which required the changing of the ending chords.

3.5.2 Experiment 2: Controlled experiments

In contrast to the first study, we designed a controlled experiment to measure how the level of adventurousness of chord recommendations and how Ripples respectively impact the novelty of students’ chord progressions and their level of satisfaction.

Task, procedure, participants

The task was to transform a given 8-chord long progression to reflect the character and mood of an image, such as Van Gogh’s Starry night. The participants were told that they would not be evaluated on how well the chord progression reflects the image, but that they can see the image as a source of inspiration.

The independent variable of this experiment was the recommendation type. We designed three variations of ChordRipple: for the current active chord, recommends

- Singleton-Typical: only single-chord substitutions that are the most typical given the context
- Singleton-Adventurous: only single-chord substitutions that are similar to current active chord but less typical in the given context
- Ripple: single-chord substitutions as in Singleton-Adventurous and also Ripple substitutions that include them as the middle chord.

Each participant was first asked to complete a tutorial that involved trying out all the functionality of ChordRipple. After completing the tutorial, each participant was asked to complete all three conditions. The order of the conditions was randomized. Each condition was paired with a different initial chord progression and image.

Similar to the first set of studies, we recruited 11 music student from local universities. One student had to leave early for class and did not complete the study. The system crashed for another student in the middle of a condition. At the end, we collected 9 complete logs.

Hypotheses

We test two hypotheses regarding how recommendation type might have an impact on the novelty of chord sequences composed and students' satisfaction.

- H1: Singleton-Atypical will result in more novel sequences (and higher satisfaction) than Singleton-Typical
- H2: Ripple will result in more novel sequences (and higher satisfaction) than Singleton-Atypical.

Design and analysis

We used a within-subjects design with one factor (recommendation type) with three levels (Singleton-Typical, Singleton-Atypical, and Ripple). We measured two variables: the user's satisfaction with the chord progressions they composed on a 5-point Likert scale, and also the novelty of their top-rated chord progression. We define novelty of a chord progression $c_{0..T-1}$ as the average of the inverse counts (as measured in the ROCK corpus) of the constituent chords, as shown in Eqn. (3.7).

$$\text{novelty}(c_{0..T-1}) = \frac{1}{T}\sum_{i=0}^{T-1}\frac{1}{N_{c_i}} \tag{3.7}$$

We used the nonparametric Friedman test to first check if there are differences among the different levels. If the test is significant, we then perform the planned pairwise comparisons following the hypotheses and use the nonparametric Wilcoxon signed-rank test to see if they are statistically significant.

Main effects and discussion

There was no effect of the recommendation type on the satisfaction of the student's ratings on their best chord progression. The means and standard deviations of the ratings in each of the conditions are shown in Table 3.3. Note that the statistical tests computed are within-subjects and the table is just for giving an overview on the distributions of students' ratings.

Measure	SingletonTypical	SingletonAtypical	Ripple
Self-rated satisfaction	4.5 (0.707)	4.06 (0.682)	4.222 (0.939)
Novelty	0.021 (0.035)	0.147 (0.091)	0.063 (0.085)

Table 3.3: The mean (standard deviation) of the two measures, self-rated satisfaction and novelty, of student's best chord sequence in the three conditions.

We did, however observe a main effect in recommendation type on the novelty of students' best-rated chord progressions. The means and standard deviations of the novelty scores in each of the conditions are shown in Table 3.3, again just to show the overall differences across conditions. The Friedman's test shows that there are differences in novelty among the three conditions within subjects, and that the effect is significant ($\chi^2(2, N = 9) = 12.67$, $p = 0.0018$). We found that when composing with the Singleton-Atypical design variation of CR, students generated chord progressions more novel than when using the Singleton-Typical design. The Wilcoxon signed-rank test shows that this effect was statistically significant ($p = 0.0039$), as shown in Table 3.4. Hence, this supports the novelty aspect of H1.

We did not observe a statistically significant effect for the pair Ripple between Singleton-Atypical ($p = 0.0977$). Hence, H2 is not supported. We also performed a post-hoc pairwise comparison and found that the Ripple condition resulted in progressions that were more novel than the Singleton-Typical and the effect was statistically significant ($p = 0.0273$).

These results show that recommending more adventurous chords can help students compose more novel chord progressions. In particular, our chord embedding model was able to make chord recommendations that help students go beyond the ordinary. We did not observe that Ripples made it easier for students to incorporate more adventurous chords.

SingletonTypical vs SingletonAtypical	SingletonTypical vs Ripple	SingleAtypical vs Ripple
$S = -22.50$, $p = 0.0039$	$S = -18.50$, $p = 0.0273$	$S = 14.50$, $p = 0.0977$

Table 3.4: Comparisons on the novelty of student's best chord sequence between all the pairs of conditions, reporting the test statistic S and two-tailed significance p from the the Wilcoxon signed-rank test.

We speculate that it is because Ripples tend to smooth things out and use less atypical chords in a row. Moreover, novelty itself does not always lead to satisfying progressions. For example, a chord progression may contain many novel chords but incoherent.

Furthermore, there are also many reasons why a Ripple might be chosen, beyond smoothing out the transitions between the original context and recommended novel chord. For example, a Ripple may be used because it is an interesting variation of the original context, or because the Ripple changes more chords at once and allows users to more quickly explore different ideas.

3.6 Broader discussion and design implications

We have observed that CR had many positive effects on composers' compositional process, although the particular outcomes depended on the person and the details of the task they were working on. Therefore, while this approach is overall helpful, a one-size-fits-all solution is not optimal.

There are several factors that affect how adventurous the composer may want the chord recommendations to be. First, a composer's musical and stylistic aspirations would influence their receptiveness to less frequently heard chords. For example, P3 described his goal as writing a "poppy" song that would be easy to relate to, and wanted to use chords that the general audience would find familiar. P5 was seeking chords that carried the sound typical of a genre. Many other students had the goal of "breaking out", and wanted to use chords that would be less likely encountered if they were improvising on their instruments.

Also, adventurousness is contextual. P8 considered using a triad among a chord sequence of mostly seventh chords as a bold move, as the triad really "stands out". At other times, the intention of the composer may be to find "something simple" so that it does not take away from the other more "iconic moments" of the phrase [P8].

We specifically designed CR to provide suggestions ranging from very typical to very adventurous, so that it could accommodate the diverse needs of composers. However, CR currently takes a fixed approach, splitting its recommendations to be half typical and half adventurous, but composers at times wanted more of one kind. Composers sometimes found some of the recommendations to be too similar to each other.

We propose that designers could include a mechanism that explicitly allows composers to adjust how typical to adventurous they want the recommendations to be. Systems should also allow composers to control the scope of diversity for recommendations, so that they could adjust between widely exploring the chord palette versus zooming into the variations of a particular sound. Furthermore, composers have different goals for different parts of a piece. For example, the intention of "making it longer" versus "I'm trying to end here" would entail very different chord recommendations. The former may include chords that open up new directions while the latter would focus on chords that provide a resolution. We propose that there could be a "knob" that allows users to control the amount of "tension" desired in the chord recommendations. It may also be possible to automatically infer a composer's intention for "conclusiveness" based on the musical context and the chords she has explored so far. The advantage of a knob is that it affords the user with more fine-grained control and the potential to explore a wider range of possibilities iteratively. The advantage of inferred intentions is that they can be used to automatically adapt the chord recommendations, allowing composers to see only what is relevant and thus making the process of composing faster.

Chapter 4

Coconet: Counterpoint by Convolution

Machine learning models of music typically break up the task of composition into a chronological process, composing a piece of music in a single pass from beginning to end. On the contrary, human composers write music in a nonlinear fashion, scribbling motifs here and there, often revisiting choices previously made. In order to better approximate this process, we train a convolutional neural network to complete partial musical scores, and explore the use of blocked Gibbs sampling as an analogue to rewriting. Neither the model nor the generative procedure are tied to a particular causal direction of composition.

Our model is an instance of orderless NADE (Uria et al., 2014), which allows more direct ancestral sampling. However, we find that Gibbs sampling greatly improves sample quality, which we demonstrate to be due to some conditional distributions being poorly modeled. Moreover, we show that even the cheap approximate blocked Gibbs procedure from Yao et al. (2014) yields better samples than ancestral sampling, based on both log-likelihood and human evaluation.

■ 4.1 Introduction

Counterpoint is the process of placing notes against notes to construct a polyphonic musical piece (Fux, 1965). This is a challenging task, as each note has strong musical influences on its neighbors and notes beyond. Human composers have developed systems of rules to guide their compositional decisions. However, these rules sometimes contradict each other, and can fail to prevent their users from going down musical dead ends. Statistical models of music, which is our current focus, is one of the many computational approaches that can help composers try out ideas more quickly, thus reducing the cost of exploration (Fernández and Vico, 2013).

Whereas previous work in statistical music modeling has relied mainly on sequence models such as Hidden Markov Models (HMMs (Baum and Petrie, 1966)) and Recurrent Neural Networks (RNNs (Rumelhart et al., 1988)), we instead employ convolutional neural networks due to their invariance properties and emphasis on capturing local structure. Nevertheless, they have also been shown to successfully model large-scale structure (van den Oord et al., 2016; van den Oord et al.). Moreover, convolutional neural networks have shown to be extremely versatile once trained, as demonstrated by a variety of creative uses such as DeepDream (Mordvintsev et al., 2015) and style transfer (Gatys et al., 2016).

We introduce COCONET, a deep convolutional model trained to reconstruct partial scores. Once trained, COCONET provides direct access to all conditionals of the form $p(\mathbf{x}_i \mid \mathbf{x}_C)$ where C selects a fragment of a musical score $\mathbf{x}$ and $i \notin C$ is in its complement. COCONET is an instance of deep orderless NADE (Uria et al., 2014), which learns an ensemble of factorizations of the joint $p(\mathbf{x})$, each corresponding to a different ordering. A related approach is the multi-prediction training of deep Boltzmann machines (MP-DBM) (Goodfellow et al., 2013), which also gives a model that can predict any subset of variables given its complement.

However, the sampling procedure for orderless NADE treats the ensemble as a mixture and relies heavily on ordering. Sampling from an orderless NADE involves (randomly) choosing an ordering, and sampling variables one by one according to the chosen ordering. This process is called *ancestral sampling*, as the order of sampling follows the directed structure of the model. We have found that this produces poor results for the highly structured and complex domain of musical counterpoint.

Instead, we propose to use blocked-Gibbs sampling, a Markov Chain Monte Carlo method to sample from a joint probability distribution by repeatedly resampling subsets of variables using conditional distributions derived from the joint probability distribution. An instance of this was previously explored by Yao et al. (2014) who employed a NADE in the transition operator for a Markov Chain, yielding a Generative Stochastic Network (GSN). The transition consists of a corruption process that masks out a subset $\mathbf{x}_{\neg C}$ of variables, followed by a process that independently resamples variables $\mathbf{x}_i$ (with $i \notin C$) according to the distribution $p_\theta(\mathbf{x}_i \mid \mathbf{x}_C)$ emitted by the model with parameters θ. Crucially, the effects of independent sampling are amortized by annealing the probability with which variables are masked out. Whereas Yao et al. (2014) treat their procedure as a cheap approximation to ancestral sampling, we find that it produces superior samples. Intuitively, the resampling process allows the model to iteratively *rewrite* the score, giving it the opportunity to correct its own mistakes.

COCONET addresses the general task of completing partial scores; special cases of this task include "bridging" two musical fragments, and temporal upsampling and extrapolation. Figure 4.1 shows an example of COCONET populating a partial piano roll using blocked-

Gibbs sampling. Code and samples are publically available.[1] Our samples on a variety of generative tasks such as rewriting, melodic harmonization and unconditioned polyphonic music generation show the versatility of our model. In this work we focus on Bach chorales, and assume four voices are active at all times. However, our model can be easily adapted to the more general, arbitrarily polyphonic representation as used in Boulanger-Lewandowski et al. (2012). Section 4.2 discusses related work in modeling music composition, with a focus on counterpoint. The details of our model and training procedure are laid out in Section 4.3. We discuss evaluation under the model in Section 4.3.2, and sampling from the model in Section 4.4. Results of quantitative and qualitative evaluations are reported in Section 4.5.

[1]

Code:	`https://github.com/czhuang/coconet`
Data:	`https://github.com/czhuang/JSB-Chorales-dataset`
Samples:	`https://coconets.github.io/`

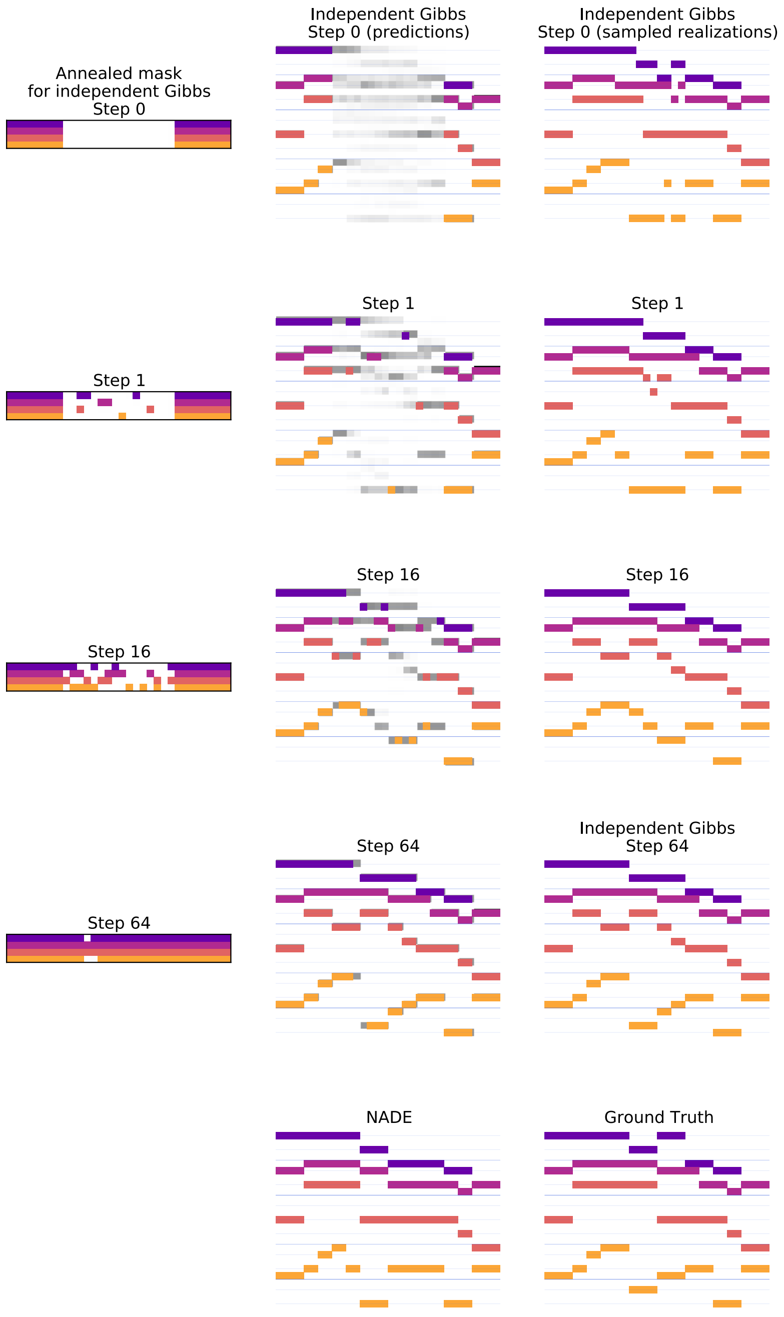


Figure 4.1: Blocked Gibbs inpainting of a corrupted Bach chorale by COCONET. At each step, a random subset of notes is removed, and the model is asked to infer their values. New values are sampled from the probability distribution put out by the model, and the process is repeated. Left: annealed masks show resampled variables. Colors distinguish the four voices. Middle: grayscale heatmaps show predictions $p(\mathbf{x}_j \mid \mathbf{x}_C)$ summed across instruments. Right: complete pianorolls after resampling the masked variables. Bottom: a sample from NADE (left) and the original Bach chorale fragment (right).

4.2 Related work

Computers have been used since their early days for experiments in music composition. A notable composition is the string quartet Illiac Suite by Hiller Jr and Isaacson (1957), which experiments with statistical sequence models such as Markov chains. One challenge in adapting such models is that music consists of multiple interdependent streams of events. Compare this to typical sequence domains such as speech and language, which involve modeling a single stream of events: a single speaker or a single stream of words. In music, extensive theories in counterpoint have been developed to address the challenge of composing multiple streams of notes that coordinate. One notable theory is due to Fux (1965) from the Baroque period, which introduces *species counterpoint* as a pedagogical scheme to gradually introduce students to the complexity of counterpoint. In first species counterpoint only one note is composed against every note in a given fixed melody (*cantus firmus*), with all notes bearing equal durations and the resulting vertical intervals consisting of only consonances.

Computer music researchers have taken inspiration from this pedagogical scheme by first teaching computers to write species counterpoint as opposed to full-fledged counterpoint. Farbood and Schöner (2001) uses Markov chains to capture transition probabilities of different melodic and harmonic transitions rules. Herremans and Sörensen (2012, 2013) takes an optimization approach by writing down an objective function that consists of existing rules of counterpoint and using a variable neighbourhood search (VNS) algorithm to optimize it.

J.S. Bach chorales has been the main corpus in computer music that serves as a starting point to tackle full-fledged counterpoint. A wide range of approaches have been used to generate music in the style of Bach chorales, for example rule-based and instance-based approaches such as the recombinancy method by Cope (1991). This method involves first segmenting existing Bach chorales into smaller chunks based on music theory, analyzing their function and stylistic signatures and then re-concatenating the chunks into new coherent works. Other approaches range from constraint-based (Pachet and Roy, 2001) to statistical methods (Conklin). In addition, Fernández and Vico (2013) gives a comprehensive survey of AI methods used not just for generating Bach chorales, but also algorithmic composition in general.

Sequence models such as HMMs and RNNs are natural choices for modeling music. Successful application of such models to polyphonic music often requires serializing or otherwise re-representing the music to fit the sequence paradigm. For instance, Liang (2016) for BachBot serializes four-part Bach chorales by interleaving the parts, while Allan and Williams (2005) construct a chord vocabulary. Boulanger-Lewandowski et al. (2012) adopt a piano roll representation, a binary matrix $\mathbf{x}$ where $\mathbf{x}_{it} = 1$ iff some instrument is playing pitch i at time t. To model the joint probability distribution of the multi-hot pitch vector $\mathbf{x}_t$, they employ a Restricted Boltzmann Machine (RBM (Smolensky, 1986; Hinton et al., 2006)) or Neural Autoregressive Distribution Estimator (NADE (Larochelle and Murray, 2011)) at each time step. Similarly Goel et al. (2014) employ a Deep Belief Network (Hinton et al., 2006) on top of an RNN.

Hadjeres et al. (2016) instead employ an undirected Markov model to learn pairwise relationships between neighboring notes up to a specified number of steps away in a score. Sampling involves Markov Chain Monte Carlo (MCMC) using the model as a Metropolis-Hastings (MH) objective. The model permits constraints on the state space to support tasks such as melody harmonization. However, the Markov assumption can limit the expressivity of the model.

Hadjeres et al. (2017) for DeepBach model note predictions by breaking down its full context into three parts, with the past and the future modeled by stacked LSTMs going in the forward and backward directions respectively, and the present harmonic context modeled by a third neural network. The three are then combined by a fourth neural network and used in Gibbs sampling for generation.

Lattner et al. (2018) imposes higher-level structure by interleaving selective Gibbs sampling on a convolutional RBM and gradient descent that minimizes cost to template piece on features such as self-similarity. This procedure itself is wrapped in simulated annealing to ensure steps do not lower the solution quality too much.

We opt for an orderless NADE training procedure which enables us to train a mixture of all possible directed models simultaneously. Finally, an approximate blocked Gibbs sampling procedure (Yao et al., 2014) allows fast generation from the model.

■ 4.3 Model

We employ machine learning techniques to obtain a generative model of musical counterpoint in the form of piano rolls. Given a dataset of observed musical pieces $\mathbf{x}^{(1)} \dots \mathbf{x}^{(n)}$ posited to come from some true distribution $p(\mathbf{x})$, we introduce a model $p_\theta(\mathbf{x})$ with parameters θ. In order to make $p_\theta(\mathbf{x})$ close to $p(\mathbf{x})$, we maximize the data log-likelihood $\sum_i \log p_\theta(\mathbf{x}^{(i)})$ (an approximation of $\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \log p_\theta(\mathbf{x})$) by stochastic gradient descent.

■ 4.3.1 Orderless NADE

The joint distribution $p(\mathbf{x})$ over D variables $\mathbf{x}_1 \dots \mathbf{x}_D$ is often difficult to model directly and hence we construct our model $p_\theta(\mathbf{x})$ from simpler factors. In the NADE (Larochelle and Murray, 2011) framework, the joint $p_\theta(\mathbf{x})$ is factorized autoregressively, one variable at a time, according to some ordering $o = o_1 \dots o_D$, such that

$$p_\theta(\mathbf{x}) = \prod_d p_\theta(\mathbf{x}_{o_d} \mid \mathbf{x}_{o_{<d}}). \tag{4.1}$$

For example, it can be factorized in chronological order:

$$p_\theta(\mathbf{x}) = p_\theta(\mathbf{x}_1)p_\theta(\mathbf{x}_2|\mathbf{x}_1)\dots p_\theta(\mathbf{x}_D|\mathbf{x}_{D-1}\dots\mathbf{x}_1) \tag{4.2}$$

In general, NADE permits any one fixed ordering, and although all orderings are equivalent from a theoretical perspective, they differ in practice due to effects of optimization and approximation. Instead, we can train NADE for all orderings o simultaneously using the orderless NADE (Uria et al., 2014) training procedure. This procedure relies on the observation that, thanks to parameter sharing, computing $p_\theta(\mathbf{x}_{o_{d'}} \mid \mathbf{x}_{o_{<d}})$ for all $d' \geq d$ is no more expensive than computing it only for $d' = d$.[2] Hence for a given o and d we can simultaneously obtain partial losses for all orderings that agree with o up to d:

$$\mathcal{L}(\mathbf{x}; o_{<d}, \theta) = -\sum_{o_d} \log p_\theta(\mathbf{x}_{o_d} \mid \mathbf{x}_{o_{<d}}, o_{<d}, o_d) \tag{4.3}$$

An orderless NADE model offers direct access to all distributions of the form $p_\theta(\mathbf{x}_i|\mathbf{x}_C)$ conditioned on any set of contextual variables $\mathbf{x}_C = \mathbf{x}_{o_{<d}}$ that might already be known. This gives us a very flexible generative model; in particular, we can use these conditional distributions to complete arbitrarily partial musical scores.

To train the model, we sample a training example $\mathbf{x}$ and context C such that $|C| \sim U(1, D)$, and update θ based on the gradient of the loss given by Equation 4.3. This loss consists of $D - d + 1$ terms, each of which corresponds to one ordering. To ensure all orderings are trained evenly we must reweight the gradients by $1/(D - d + 1)$. This correction, due to Uria et al. (2014), ensures consistent estimation of the joint negative log-likelihood $\log p_\theta(\mathbf{x})$.

In this work, the model $p_\theta(x)$ is implemented by a deep convolutional neural network (Krizhevsky et al., 2012). We represent the music as a stack of piano rolls encoded in a

[2]Here $\mathbf{x}_{o_{<d}}$ is used as shorthand for variables $\mathbf{x}_{o_1} \dots \mathbf{x}_{o_{d-1}}$ that occur earlier in the ordering.

binary three-dimensional tensor $\mathbf{x} \in \{0,1\}^{I \times T \times P}$. Here I denotes the number of instruments, T the number of time steps, P the number of pitches, and $\mathbf{x}_{i,t,p} = 1$ iff the ith instrument plays pitch p at time t. We will assume each instrument plays exactly one pitch at a time, that is, $\sum_p \mathbf{x}_{i,t,p} = 1$ for all i, t.

Our focus is on four-part Bach chorales as used in prior work (Allan and Williams, 2005; Boulanger-Lewandowski et al., 2012; Goel et al., 2014; Liang, 2016; Hadjeres et al., 2016). Hence we assume $I = 4$ throughout. We constrain ourselves to only the range that appears in our training data (MIDI pitches 36 through 88). Time is discretized at the level of 16th notes for similar reasons. To curb memory requirements, we enforce $T = 128$ by randomly cropping the training examples.

Given a training example $\mathbf{x} \sim p(\mathbf{x})$, we present the model with values of only a strict subset of its elements $\mathbf{x}_C = \{\mathbf{x}_{(i,t)} \mid (i,t) \in C\}$ and ask it to reconstruct its complement $\mathbf{x}_{\neg C}$. The input $\mathbf{h}^0 \in \{0,1\}^{2I \times T \times P}$ is obtained by masking the piano rolls $\mathbf{x}$ to obtain the context $\mathbf{x}_C$ and concatenating this with the corresponding mask:

$$\mathbf{h}^0_{i,t,p} = \mathbb{1}_{(i,t) \in C} \mathbf{x}_{i,t,p} \tag{4.4}$$

$$\mathbf{h}^0_{I+i,t,p} = \mathbb{1}_{(i,t) \in C} \tag{4.5}$$

where the time and pitch dimensions are treated as spatial dimensions to convolve over. Each instrument's piano roll $\mathbf{h}^0_i$ and mask $\mathbf{h^0_{I+i}}$ is treated as a separate channel and convolved independently.

With the exception of the first and final layers, all convolutions preserve the size of the hidden representation. That is, we use "same" padding throughout and all activations have the same number of channels H, such that $\mathbf{h}^l \in \mathbb{R}^{H \times T \times P}$ for all $1 < l < L$. Throughout our experiments we used $L = 64$ layers and $H = 128$ channels. After each convolution we apply batch normalization (Ioffe and Szegedy, 2015) (denoted by $\mathrm{BN}(\cdot)$) with statistics tied across time and pitch. Batch normalization rescales activations at each layer to have mean β and standard deviation γ, which greatly improves optimization. After every second convolution, we introduce a skip connection from the hidden state two levels below to reap the benefits of residual learning (He et al., 2016).

$$\mathbf{a}^l = \mathrm{BN}(\mathbf{W}^l * \mathbf{h}^{l-1}; \gamma^l, \beta^l) \tag{4.6}$$

$$\mathbf{h}^l = \begin{cases} \mathrm{ReLU}(\mathbf{a}^l + \mathbf{h}^{l-2}) \\ \qquad\qquad \text{if } 3 < l < L-1 \text{ and } l \bmod 2 = 0 \\ \mathrm{ReLU}(\mathbf{a}^l) \text{ otherwise} \end{cases}$$

$$\mathbf{h}^L = \mathbf{a}^L \tag{4.7}$$

The final activations $\mathbf{h}^L \in \mathbb{R}^{I \times T \times P}$ are passed through the softmax function to obtain predictions for the pitch at each instrument/time pair:

$$p_\theta(\mathbf{x}_{i,t,p} \mid \mathbf{x}_C, C) = \frac{\exp(h^L_{i,t,p})}{\sum_p \exp(h^L_{i,t,p})} \tag{4.8}$$

The loss function from Equation 4.3 is then given by

$$\begin{aligned} \mathcal{L}(\mathbf{x}; C, \theta) &= -\sum_{(i,t) \notin C} \log p_\theta(\mathbf{x}_{i,t} \mid \mathbf{x}_C, C) \\ &= -\sum_{(i,t) \notin C} \sum_p \mathbf{x}_{i,t,p} \log p_\theta(\mathbf{x}_{i,t,p} \mid \mathbf{x}_C, C) \end{aligned} \tag{4.9}$$

where p_θ denotes the probability under the model with parameters $\theta = \mathbf{W}^1, \gamma^1, \beta^1, \ldots, \mathbf{W}^{L-1}, \gamma^{L-1}, \beta^{L-1}$. We train the model by minimizing

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \mathbb{E}_{C \sim p(C)} \frac{1}{|\neg C|} \mathcal{L}(\mathbf{x}; C, \theta) \tag{4.10}$$

with respect to θ using stochastic gradient descent with step size determined by Adam (Kingma and Ba, 2014). The expectations are estimated by sampling piano rolls $\mathbf{x}$ from the training set and drawing a single context C per sample.

■ 4.3.2 Loss formulation

The log-likelihood of a given example is computed according to Algorithm 2 by repeated application of Equation 4.8. Evaluation occurs one frame at a time, within which the model conditions on its own predictions and does not see the ground truth. Unlike notewise teacher-forcing, where the ground truth is injected after each prediction, the framewise evaluation is thus sensitive to accumulation of error. This gives a more representative measure of quality of the generative model. For each example, we repeat the evaluation process a number of times to average over multiple orderings, and finally average across frames and examples. For chronological evaluation, we draw only orderings that have the t_ls in increasing order.

Algorithm 2 Framewise log-likelihood evaluation

```
Given a piano roll $\mathbf{x}$
$L_{m,i,t} \leftarrow 0$ for all $m, i, t$
for multiple orderings $m = 0 \ldots M$ do
    $C \leftarrow \emptyset$, $\widehat{\mathbf{x}} \leftarrow \mathbf{x}$
    Sample an ordering $t_1, t_2 \ldots t_T$ over frames
    for $l = 0 \ldots T$ do
        Sample an ordering $i_1, i_2 \ldots i_I$ over instruments
        for $k = 0 \ldots I$ do
            $\pi_p \leftarrow p_\theta(\mathbf{x}_{i_k,t_l,p} \mid \widehat{\mathbf{x}}_C, C)$ for all $p$
            $L_{m,i_k,t_l} \leftarrow \sum_p \mathbf{x}_{i_k,t_l,p} \log \pi_p$
            $\widehat{\mathbf{x}}_{i_k,t_l} \sim \mathrm{Cat}(P, \pi)$
            $C \leftarrow C \cup (i_k, t_l)$
        end for
        $\widehat{\mathbf{x}}_C \leftarrow \mathbf{x}_C$
    end for
end for
return $-\frac{1}{T} \sum_t \log \frac{1}{M} \sum_m \exp \sum_i L_{m,i,t}$
```

4.4 Sampling

We can sample from the model using the orderless Nade ancestral sampling procedure, in which we first sample an ordering and then sample variables one by one according to the ordering. However, we find that this yields poor samples, and we propose instead to use Gibbs sampling.

4.4.1 Orderless NADE sampling

Sampling according to orderless Nade involves first randomly choosing an ordering and then sampling variables one by one according to the chosen ordering. We use an equivalent procedure in which we arrive at a random ordering by at each step randomly choosing the next variable to sample. We start with an empty (zero everywhere) piano roll $\mathbf{x}^0$ and empty context C^0 and populate them iteratively by the following process. We feed the piano roll $\mathbf{x}^s$ and context C^s into the model to obtain a set of categorical distributions

$p_\theta(\mathbf{x}_{i,t}|\mathbf{x}^s_{C^s}, C^s)$ for $(i,t) \notin C^s$. As the $\mathbf{x}_{i,t}$ are not conditionally independent, we cannot simply sample from these distributions independently. However, if we sample from one of them, we can compute new conditional distributions for the others. Hence we randomly choose one $(i,t)^{s+1} \notin C^s$ to sample from, and let $\mathbf{x}^{s+1}_{i,t}$ equal the one-hot realization. Augment the context with $C^{s+1} = C^s \cup (i,t)^{s+1}$ and repeat until the piano roll is populated. This procedure is easily generalized to tasks such as melody harmonization and partial score completion by starting with a nonempty piano roll.

Unfortunately, samples thus generated are of low quality, which we surmise is due to accumulation of errors. This is a well-known weakness of autoregressive models. Venkatraman et al. (2015); Bengio et al. (2015); Huszár (2015); Lamb et al. (2016) While the model provides conditionals $p_\theta(\mathbf{x}_{i,t}|\mathbf{x}_C, C)$ for all $(i,t) \notin C$, some of these conditionals may be better modeled than others. We suspect in particular those conditionals used early on in the procedure, for which the context C consists of very few variables. Moreover, although the model is trained to be order-agnostic, different orderings invoke different distributions, which is another indication that some conditionals are poorly learned. We test this hypothesis in Section 4.5.2.

■ 4.4.2 Gibbs sampling

To remedy this, we allow the model to revisit its choices: we repeatedly mask out some part of the piano roll and then repopulate it. This is a form of blocked Gibbs sampling Liu (1994). Blocked sampling is crucial for mixing, as the high temporal resolution of our representation causes strong correlations between consecutive notes. For instance, without blocked sampling, it would take many steps to snap out of a long-held note. Similar considerations hold for the Ising model from statistical mechanics, leading to the Swendsen-Wang algorithm Swendsen and Wang (1987) in which large clusters of variables are resampled at once.

We consider two strategies for resampling a given block of variables: *ancestral* sampling and *independent* sampling. Ancestral sampling invokes the orderless NADE sampling procedure described in Section 4.4.1 on the masked-out portion of the piano roll. Independent sampling simply treats the masked-out variables $\mathbf{x}_{\neg C}$ as independent given the context $\mathbf{x}_C$.

Using independent blocked Gibbs to sample from a NADE model has been studied by Yao et al. (2014), who propose to use an annealed masking probability $\alpha_n = \max(\alpha_{\min}, \alpha_{\max} - n(\alpha_{\max} - \alpha_{\min})/(\eta N))$ for some minimum and maximum probabilities $\alpha_{\min}, \alpha_{\max}$, total number of Gibbs steps N and fraction η of time spent before settling onto the minimum probability $\alpha_{\min}$. Initially, when the masking probability is high, the chain mixes fast but samples are poor due to independent sampling. As α_n decreases, the blocked Gibbs process with independent resampling approaches standard Gibbs where one variable at a time is resampled, thus amortizing the effects of independent sampling. N is a hyperparameter which as a rule of thumb we set equal to IT; it can be set lower than that to save computation at a slight loss of sample quality.

Yao et al. (2014) treat independent blocked Gibbs as a cheap approximation to ancestral sampling. Whereas plain ancestral sampling (4.4.1) requires $O(IT)$ model evaluations, ancestral blocked Gibbs requires a prohibitive $O(ITN)$ model evaluations and independent Gibbs requires only $O(N)$, where N can be chosen to be less than IT. Moreover, we find that independent blocked Gibbs sampling in fact yields *better* samples than plain ancestral sampling.

4.5 Experiments

We evaluate our approach on a popular corpus of four-part Bach chorales. While the literature features many variants of this dataset Allan and Williams (2005); Boulanger-Lewandowski et al. (2012); Liang (2016); Hadjeres et al. (2016), we report results on that used by Boulanger-Lewandowski et al. (2012). As the quarter-note temporal resolution used by Boulanger-Lewandowski et al. (2012) is frankly too coarse to accurately convey counterpoint, we also evaluate on eighth-note and sixteenth-note quantizations of the same data.

It should be noted that quantitative evaluation of generative models is fundamentally hard Theis et al. (2016). The gold standard for evaluation is qualitative comparison by humans, and we therefore report human evaluation results as well.

4.5.1 Data log-likelihood

Table 4.1 compares the framewise log-likelihood of the test data under variants of our model and those reported in Boulanger-Lewandowski et al. (2012). We find that the temporal resolution has a dramatic influence on the performance, which we suspect is an artifact of the performance metric. The log-likelihood is evaluated by teacher-forcing, that is, the prediction of a frame is conditioned on the ground truth of all previously predicted frames. As temporal resolution increases, chord changes become increasingly rare, and the model is increasingly rewarded for simply holding notes over time.

We evaluate Coconet on both chronological and random orderings, in both cases averaging likelihoods across an ensemble of $M = 5$ orderings. The chronological orderings differ only in the ordering of instruments within each frame. We see in Table 4.1 that fully random orderings lead to significantly better performance. We believe the members of the more diverse random ensemble are more mutually complementary. For example, a forward ordering is uncertain at the beginning of a piece and more certain toward the end, whereas a backward ordering is more certain at the beginning and less certain toward the end.

4.5.2 Sample quality

In Section 4.4 we conjectured that the low quality of NADE samples is due to poorly modeled conditionals $p_\theta(\mathbf{x}_{i,t} \mid \mathbf{x}_C, C)$ where C is small. We test this hypothesis by evaluating the likelihood under the model of samples generated by the ancestral blocked Gibbs procedure with C chosen according to independent Bernoulli variables. When we set the inclusion probability ρ to 0, we obtain NADE. Increasing ρ increases the expected context size $|C|$, which should yield better samples if our hypothesis is true. The results shown in Table 4.2 confirm that this is the case. For these experiments, we used sample length $T = 32$ time steps and number of Gibbs steps $N = 100$.

Figure 4.2 shows the convergence behavior of the various Gibbs procedures, averaged over 100 runs. We see that for low values of ρ (small C), the chains hardly make progress beyond NADE in terms of likelihood. Higher values of ρ (large C) enable the model to get off the ground and reach significantly better likelihood.

Model	**Temporal resolution**		
	quarter	eighth	sixteenth
NADE Boulanger-Lewandowski et al. (2012)	7.19		
RNN-RBM Boulanger-Lewandowski et al. (2012)	6.27		
RNN-NADE Boulanger-Lewandowski et al. (2012)	5.56		
RNN-NADE (our implementation)	5.03	3.78	2.05
COCONET (chronological)	7.79 ± 0.09	4.21 ± 0.05	2.22 ± 0.03
COCONET (random)	5.03 ± 0.06	1.84 ± 0.02	0.57 ± 0.01

Table 4.1: Framewise negative log-likelihoods (NLLs) on the Bach corpus. We compare against Boulanger-Lewandowski et al. (2012), who used quarter-note resolution. We also compare on higher temporal resolutions (eighth notes, sixteenth notes), against our own reimplementation of RNN-NADE. COCONET is an instance of orderless NADE, and as such we evaluate it on random orderings. However, the baselines support only chronological frame ordering, and hence we evaluate our model in this setting as well.

Sampling scheme	Framewise NLL
Ancestral Gibbs, $\rho = 0.00$ (NADE)	1.09 ± 0.06
Ancestral Gibbs, $\rho = 0.05$	1.08 ± 0.06
Ancestral Gibbs, $\rho = 0.10$	0.97 ± 0.05
Ancestral Gibbs, $\rho = 0.25$	0.80 ± 0.04
Ancestral Gibbs, $\rho = 0.50$	0.74 ± 0.04
Independent Gibbs Yao et al. (2014)	0.52 ± 0.01

Table 4.2: Mean (± SEM) NLL under model of unconditioned samples generated from model by various schemes.

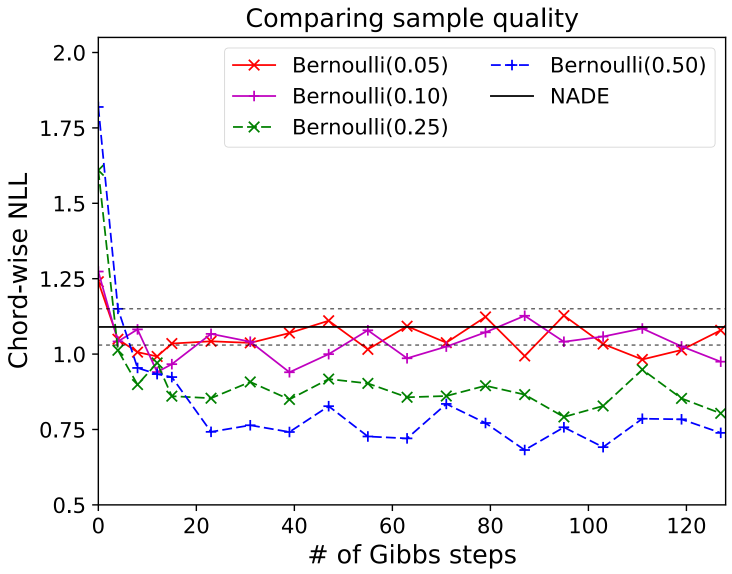


Figure 4.2: Likelihood under the model for ancestral Gibbs samples obtained with various context distributions $p(C)$. NADE (Bernoulli(0.00)) is included for reference.

4.5.3 Human evaluations

To further compare the sample quality of different sampling procedures, we carried out a listening test on Amazon's Mechanical Turk (MTurk). The procedures include orderless NADE ancestral sampling and independent Gibbs Yao et al. (2014), with each we generate four unconditioned samples of eight-measure lengths from empty piano rolls. To have an absolute reference for the quality of samples, we include first eight measures of four random Bach chorale pieces from the validation set. Each fragment lasts thirty-four seconds after synthesis.

For each MTurk hit, participants are asked to rate on a Likert scale which of the two random samples they perceive as more Bach-like. A total of 96 ratings were collected, with each source involved in 64 (=96*2/3) pairwise comparisons. Figure 4.3 shows the number

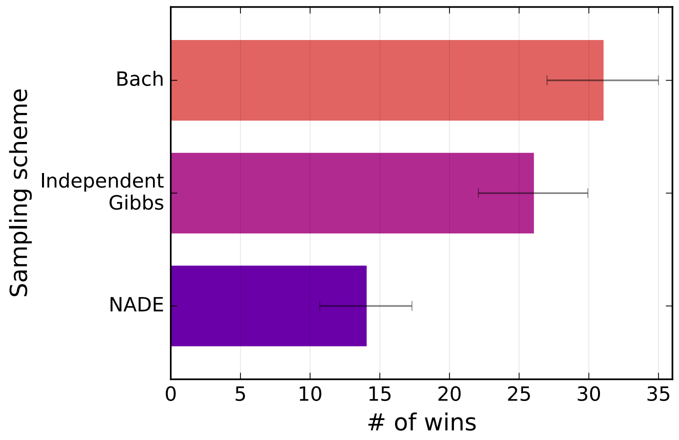


Figure 4.3: Human evaluations from MTurk on how many times a sampling procedure or Bach is perceived as more Bach-like. Error bars show the standard deviation of a binomial distribution fitted to each's binary win/loss counts.

of times each source was perceived as closer to Bach's style. We perform a Kruskal-Wallis H test on the ratings, $\chi^2(2) = 12.23, p < 0.001$, showing there are statistically significant differences between models. A post-hoc analysis using the Wilcoxon signed-rank test with Bonferroni correction showed that participants perceived samples from independent Gibbs as more Bach-like than ancestral sampling (NADE), $p < 0.05/3$. This confirms the loglikelihood comparisons on sample quality in 4.5.2 that independent Gibbs produces better samples. There was also a significance difference between Bach and ancestral samples but not between Bach and independent Gibbs.

■ 4.6 Conclusion

We introduced a convolutional approach to modeling musical scores based on the orderless NADE Uria et al. (2016) framework. Our experiments show that the NADE ancestral sampling procedure yields poor samples, which we have argued is because some conditionals are not captured well by the model. We have shown that sample quality improves significantly when we use blocked Gibbs sampling to iteratively rewrite parts of the score. Moreover, annealed independent blocked Gibbs sampling as proposed by Yao et al. (2014) is not only faster but in fact produces better samples.

Chapter 5

Music Transformer: Self-attention based music generation

Music relies heavily on repetition to build structure and meaning. Self-reference occurs on multiple timescales, from motifs to phrases to reusing of entire sections of music, such as in pieces with ABA structure. The Transformer (Vaswani et al., 2017), a sequence model based on self-attention, has achieved compelling results in many generation tasks that require maintaining long-range coherence. This suggests that self-attention might also be well-suited to modeling music. In musical composition and performance, however, relative timing is critically important. Existing approaches for representing relative positional information in the Transformer modulate attention based on pairwise distance (Shaw et al., 2018). This is impractical for long sequences such as musical compositions because their memory complexity for intermediate relative information is quadratic in the sequence length. We propose an algorithm that reduces their intermediate memory requirement to linear in the sequence length. This enables us to demonstrate that a Transformer with our modified relative attention mechanism can generate minute-long compositions (thousands of steps) with compelling structure, generate continuations that coherently elaborate on a given motif, and in a seq2seq setup generate accompaniments conditioned on melodies[1]. We evaluate the Transformer with our relative attention mechanism on two datasets, JSB Chorales and Maestro, and obtain state-of-the-art results on the latter.

5.1 Introduction

A musical piece often consists of recurring elements at various levels, from motifs to phrases to sections such as verse-chorus. To generate a coherent piece, a model needs to reference

[1]Samples are available for listening at
`https://goo.gl/magenta/music-transformer-examples`.
Later blog post at `https://magenta.tensorflow.org/music-transformer`

elements that came before, sometimes in the distant past, and then repeat, vary, and further develop them to create contrast and surprise. Intuitively, self-attention (Parikh et al., 2016) could be a good match for this task. Self-attention over its own previous outputs allows an autoregressive model to access any part of the previously generated output at every step of generation. By contrast, recurrent neural networks have to learn to proactively store elements to be referenced in a fixed size state or memory, making training potentially much more difficult. We believe that repeating self-attention in multiple, successive layers of a Transformer decoder (Vaswani et al., 2017) can help capture the multiple levels at which self-referential phenomena exist in music.

In its original formulation, the Transformer relies on absolute position representations, using either positional sinusoids or learned position embeddings that are added to the per-position input representations. Recurrent and convolutional neural networks instead model position in relative terms: RNNs through their recurrence over the positions in their input, and CNNs by applying kernels that effectively choose which parameters to apply based on the relative position of the covered input representations.

Music has multiple dimensions along which relative differences arguably matter more than their absolute values; the two most prominent are timing and pitch. To capture such pairwise relations between representations, Shaw et al. (2018) introduce a relation-aware version of self-attention which they use successfully to modulate self-attention by the distance between two positions. We extend this approach to capture relative timing and optionally also pitch, which yields improvement in both sample quality and perplexity for the JSB Chorales dataset. As opposed to the original Transformer, samples from a Transformer with our relative attention mechanism maintain the regular timing grid present in this dataset. The model furthermore captures global timing, giving rise to regular phrases.

The original formulation of relative attention (Shaw et al., 2018) requires $O(L^2D)$ memory where L is the sequence length and D is the dimension of the model's hidden state. This is prohibitive for long sequences such as those found in the Maestro dataset of human-performed virtuosic, classical piano music (Hawthorne et al., 2019). In Section 5.3.4, we show how to reduce the memory requirements to $O(LD)$, making it practical to apply relative attention to long sequences.

The Maestro dataset consists of MIDI recorded from performances of competition participants, bearing expressive dynamics and timing on a less than 10-millisecond granularity. Discretizing time in a fixed grid on such a resolution would yield unnecessarily long sequences as not all events change on the same timescale. We hence adopt a sparse, MIDI-like, event-based representation from (Oore et al., 2018), allowing a minute of music with a 10-millisecond resolution to be represented at lengths around 2K. This is in contrast to a 6K to 18K length that would be needed on a serialized multi-attribute fixed-grid representation. As position in sequence no longer corresponds to time, a priori it is not obvious that relative attention should work as well with such a representation. However, we will show in Section 5.4.3 that it does improve perplexity and sample quality over strong baselines.

We speculate that idiomatic piano gestures such as scales, arpeggios and other motifs all exhibit a certain grammar and recur periodically, hence knowing their relative positional distances makes it easier to model this regularity. This inductive bias towards learning relational information, as opposed to patterns based on absolute position, suggests that the Transformer with relative attention could generalize beyond the lengths it was trained on, which our experiments in Section 5.4.3 confirm.

5.1.1 Contributions

Domain contributions We show the first successful use of Transformers in generating music that exhibits long-term structure. Before our work, LSTMs were used at timescales of 15s (~500 tokens) of piano performances (Oore et al., 2018). Our work demonstrates that Transformers not only achieve state-of-the-art perplexity on modeling these complex expressive piano performances, but can also generate them at the scale of minutes (thousands of tokens) with remarkable internal consistency. Our relative self-attention formulation is essential to the model's quality. In listening tests (see Section 5.4.4), samples from models with relative self-attention were perceived as more coherent than the baseline Transformer model (Vaswani et al., 2017). Relative attention not only enables Transformers to generate continuations that elaborate on a given motif, but also to generalize and generate in consistent fashion beyond the length it was trained on (see Section 5.4.3). In a seq2seq setup, Transformers can generate accompaniments conditioned on melodies, enabling users to interact with the model.

Algorithmic contributions The space complexity of the relative self-attention mechanism in its original formulation (Shaw et al., 2018) made it infeasible to train on sequences of sufficient length to capture long-range structure in longer musical compositions. To address this, we present a crucial algorithmic improvement to the relative self-attention mechanism, dramatically reducing its memory requirements from $O(L^2D)$ to $O(LD)$. For example, the memory consumption per layer is reduced from 8.5 GB to 4.2 MB (per head from 1.1 GB to 0.52 MB) for a sequence of length $L = 2048$ and hidden-state size $D = 512$ (per head $D_h = \frac{D}{H} = 64$, where number of heads is $H = 8$) (see Table 5.1), allowing us to use GPUs to train the relative self-attention Transformer on long sequences.

5.2 Related work

Sequence models have been the canonical choice for modeling music, from Hidden Markov Models to RNNs and Long Short Term Memory networks (e.g., Eck and Schmidhuber, 2002; Liang, 2016; Oore et al., 2018), to bidirectional LSTMs (e.g., Hadjeres et al., 2017). Successful application of sequential models to polyphonic music often requires serializing the musical score or performance into a single sequence, for example by interleaving different instruments or voices. Alternatively, a 2D pianoroll-like representation (see 5.4.1 for more details) can be decomposed into a sequence of multi-hot pitch vectors, and their joint probability distributions can be captured using Restricted Boltzmann Machines (Smolensky, 1986; Hinton et al., 2006) or Neural Autoregressive Distribution Estimators (NADE; Larochelle and Murray, 2011). Pianorolls are also image-like and can be modeled by CNNs trained either as generative adversarial networks (GAN; Goodfellow et al., 2014). (e.g., Dong et al., 2018) or as orderless NADEs (Uria et al., 2014, 2016) (e.g., Huang et al., 2017).

Lattner et al. (2018) use self-similarity in style-transfer fashion, where the self-similarity structure of a piece serves as a template objective for gradient descent to impose similar repetition structure on an input score. Self-attention can be seen as a generalization of self-similarity; the former maps the input through different projections to queries and keys, and the latter uses the same projection for both.

Dot-product self-attention is the mechanism at the core of the Transformer, and several recent works have focused on applying and improving it for image generation, speech, and summarization (Parmar et al., 2018; Povey et al., 2018; Liu et al., 2018). A key challenge encountered by each of these efforts is scaling attention computationally to long sequences. This is because the time and space complexity of self-attention grows quadratically in the sequence length. For relative self-attention (Shaw et al., 2018) this is particularly problematic as the space complexity also grows linearly in the dimension, or depth, of the per-position representations.

5.3 Model

5.3.1 Data representations

We take a language-modeling approach to training generative models for symbolic music. Hence we represent music as a sequence of discrete tokens, with the vocabulary determined by the dataset. Datasets in different genres call for different ways of serializing polyphonic music into a single stream and also discretizing time.

The JSB Chorale dataset consists of four-part scored choral music, which can be represented as a matrix where rows correspond to voices and columns to time discretized to sixteenth notes. The matrix's entries are integers that denote which pitch is being played. This matrix can than be serialized in raster-scan fashion by first going down the rows and then moving right through the columns (see 5.4.1 for more details). Compared to JSB Chorale, the piano performance data in the Maestro dataset includes expressive timing information at much finer granularity and more voices. For the Maestro dataset we therefore use the performance encoding proposed by Oore et al. (2018) which consists of a vocabulary of 128 `NOTE_ON` events, 128 `NOTE_OFF`s, 100 `TIME_SHIFT`s allowing for expressive timing at 10ms and 32 `VELOCITY` bins for expressive dynamics (see 5.4.1 for more details).

■ 5.3.2 Background: Self-attention in Transformer

The Transformer decoder is a autoregressive generative model that uses primarily self-attention mechanisms, and learned or sinusoidal position information. Each layer consists of a self-attention sub-layer followed by a feedforward sub-layer.

The attention layer first transforms a sequence of L D-dimensional vectors $X = (x_1, x_2, \ldots, x_L)$ into queries $Q = XW^Q$, keys $K = XW^K$, and values $V = XW^V$, where W^Q, W^K, and W^V are each $D \times D$ square matrices. Each $L \times D$ query, key, and value matrix is then split into H $L \times D_h$ parts or attention heads, indexed by h, and with dimension $D_h = \frac{D}{H}$, which allow the model to focus on different parts of the history. The scaled dot-product attention computes a sequence of vector outputs for each head as

$$Z^h = \text{Attention}(Q^h, K^h, V^h) = \text{Softmax}\left(\frac{Q^h K^{h\top}}{\sqrt{D_h}}\right) V^h. \tag{5.1}$$

The attention outputs for each head are concatenated and linearly transformed to get Z, a L by D dimensional matrix. A upper triangular mask ensures that queries cannot attend to keys later in the sequence. For other details of the Transfomer model, such as residual connections and learning rates, the reader can refer Vaswani et al. (2017). The feedforward (FF) sub-layer then takes the output Z from the previous attention sub-layer, and performs two layers of point-wise dense layers on the depth D dimension, as shown in Equation 5.2.

W_1, W_2, b_1, b_2 are weights and biases of those two layers.

$$\mathrm{FF}(Z) = \mathrm{ReLU}(ZW_1 + b_1)W_2 + b_2 \tag{5.2}$$

■ 5.3.3 Relative positional self-attention

As the Transformer model relies solely on positional sinusoids to represent timing information, Shaw et al. (2018) introduced relative position representations to allow attention to be informed by how far two positions are apart in a sequence. This involves learning a separate relative position embedding E^r of shape (H, L, D_h), which has an embedding for each possible pairwise distance $r = j_k - i_q$ between a query and key in position i_q and j_k respectively. The embeddings are ordered from distance $-L + 1$ to 0, and are learned separately for each head. In Shaw et al. (2018), the relative embeddings interact with queries and give rise to a S^{rel}, an $L \times L$ dimensional logits matrix which modulates the attention probabilities for each head as:

$$\mathrm{RelativeAttention} = \mathrm{Softmax}\left(\frac{QK^\top + S^{rel}}{\sqrt{D_h}}\right)V. \tag{5.3}$$

We dropped head indices for clarity. Our work uses the same approach to infuse relative distance information in the attention computation, while significantly improving upon the memory footprint for computing S^{rel}. For each head, Shaw et al. (2018) instantiate an intermediate tensor R of shape (L, L, D_h), containing the embeddings that correspond to the relative distances between all keys and queries. Q is then reshaped to an $(L, 1, D_h)$ tensor, and $S^{rel} = QR^\top$.[2] This incurs a total space complexity of $O(L^2 D)$, restricting its application to long sequences.

■ 5.3.4 Memory efficient implementation of relative position-based attention

We improve the implementation of relative attention by reducing its intermediate memory requirement from $O(L^2 D)$ to $O(LD)$, with example lengths shown in Table 5.1. We observe that all of the terms we need from $QR^\top$ are already available if we directly multiply Q with E^r, the relative position embedding. After we compute $QE^{r\top}$, its (i_q, r) entry contains the dot product of the query in position i_q with the embedding of relative distance r. However, each relative logit (i_q, j_k) in the matrix S^{rel} from Equation 5.3 should be the dot product of the query in position i_q and the embedding of the relative distance $j_k - i_q$, to match up with

[2]We assume that the batch size is 1 here. With a batch size of B, Q would be reshaped to (L, B, D_h) and S^{rel} would be computed with a batch matrix–matrix product.

the indexing in $QK^\top$. We therefore need to "skew" $QE^{r\top}$ so as to move the relative logits to their correct positions, hence $S^{rel} = \text{Skew}(QE^r)$. The "skewing" procedure is illustrated in Figure 5.1 and will be detailed in the next section. The time complexity for both methods are $O(L^2D)$, while in practice our method is 6x faster at length 650 as prior work still requires manipulating larger tensors.

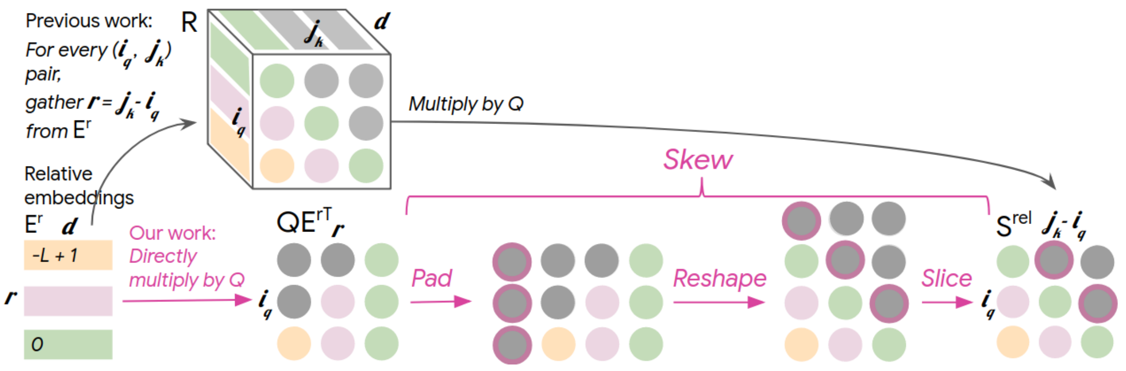


Figure 5.1: Relative global attention: the bottom row describes our memory-efficient "skewing" algorithm, which does not require instantiating R (top row, which is $O(L^2D)$). Gray indicates masked or padded positions. Each color corresponds to a different relative distance.

Table 5.1: Comparing the overall relative memory complexity (intermediate relative embeddings (R or E^r) + relative logits S^{rel}), the maximal training lengths that can fit in a GPU with 16GB memory assuming $D_h = 64$, and the memory usage per layer per head (in MB).

Implementation	Relative memory	Maximal L	$L = 650$	$L = 2048$	$L = 3500$
Shaw et al. (2018)	$O(L^2D + L^2)$	650	108 + 1.7	1100 + 16	3100 + 49
Ours	$O(LD + L^2)$	3500	0.17 + 1.7	0.52 + 16	0.90 + 49

The "skewing" procedure

Hence, we propose a "skewing" procedure to transform an absolute-by-relative (i_q, r) indexed matrix into an absolute-by-absolute (i_q, j_k) indexed matrix. The row indices i_q stay the same while the columns indices are shifted according to the following equation: $j_k = r - (L-1) + i_q$. For example in Figure 5.1 the upper right green dot in position $(0, 2)$ of $QE^{r\top}$ after skewing has a column index of $2 - (3 - 1) + 0 = 0$, resulting in a position of $(0, 0)$ in S^{rel}.

We outline the steps illustrated in Figure 5.1 below.

1. Pad a dummy column vector of length L before the leftmost column.

2. Reshape the matrix to have shape $(L+1, L)$. (This step assumes NumPy-style row-major ordering.)

3. Slice that matrix to retain only the last l rows and all the columns, resulting in a (L, L) matrix again, but now absolute-by-absolute indexed, which is the S^{rel} that we need.

5.3.5 Relative local attention

For very long sequences, the quadratic memory requirement of even baseline Transformer is impractical. Local attention has been used for example in Wikipedia and image generation (Liu et al., 2018; Parmar et al., 2018) by chunking the input sequence into non-overlapping blocks. Each block then attends to itself and the one before, as shown by the smaller thumbnail on the top right corner of Figure 5.2.

To extend relative attention to the local case, we first note that the right block has the same configuration as in the global case (see Figure 5.1) but much smaller: $(\frac{L}{M})^2$ (where M is the number of blocks, and N be the resulting block length) as opposed to L^2. The left block is unmasked with relative indices running from -1 (top right) to -2$N+1$ (bottom left). Hence, the learned E^r for the local case has shape $(2N-1, N)$.

Similar to the global case, we first compute $QE^{r\top}$ and then use the following procedure to skew it to have the same indexing as $QK^\top$, as illustrated in Figure 5.2.

1. Pad a dummy column vector of length N after the rightmost column.

2. Flatten the matrix and then pad with a dummy row of length $N-1$.

3. Reshape the matrix to have shape $(N+1, 2N-1)$.

4. Slice that matrix to retain only the first N rows and last N columns, resulting in a (N, N) matrix.

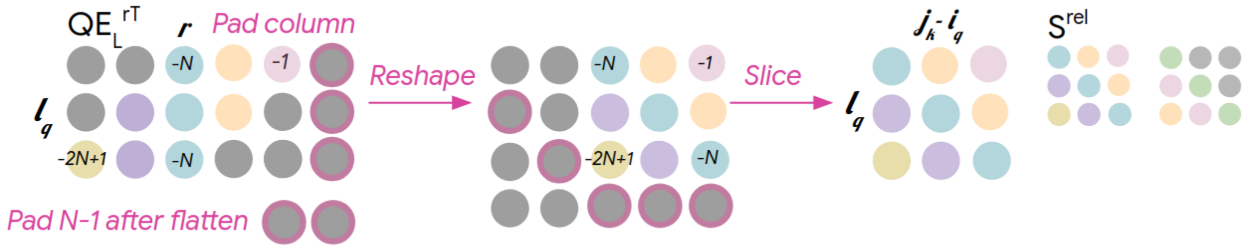


Figure 5.2: Relative local attention: the thumbnail on the right shows the desired configuration for S^{rel}. The “skewing” procedure is shown from left to right.

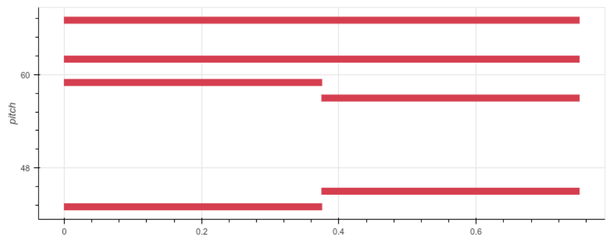
5.4 Experiments

5.4.1 Domain-specific representations

Adapting sequence models for music requires making decisions on how to serialize a polyphonic texture. The data type, whether score or performance, makes certain representations more natural for encoding all the information needed while still resulting in reasonable sequence lengths.

Serialized instrument/time grid (used for the J.S.Bach Chorales dataset)

The first dataset, J.S. Bach Chorales, consists of four-part score-based choral music. The time resolution is sixteenth notes, making it possible to use a serialized grid-like representation. Figure 5.3 shows how a pianoroll (left) can be represented as a grid (right), following (Huang et al., 2017). The rows show the MIDI pitch number of each of the four voices, from top to bottom being soprano (S), alto (A), tenor (T) and bass (B), while the columns is discretized time, advancing in sixteenth notes. Here longer notes such as quarter notes are broken down into multiple repetitions. To serialize the grid into a sequence, we interleave the parts by first iterating through all the voices at time step 1, and then move to the next column, and then iterate again from top to bottom, and so on. The resulting sequence is $S_1A_1T_1B_1S_2A_2T_2B_2...$, where the subscript gives the time step. After serialization, the most common sequence length is 1024. Each token is represented as onehot in pitch.



```
S: 67, 67, 67, 67
A: 62, 62, 62, 62
T: 59, 59, 57, 57
B: 43, 43, 45, 45
```

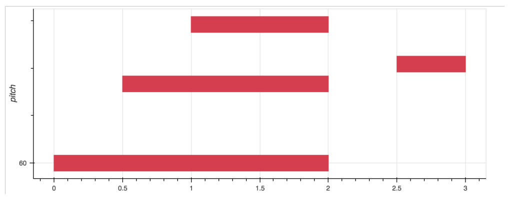
Figure 5.3: The opening measure of BWV 428 is visualized as a pianoroll (left, where the x-axis is discretized time and y-axis is MIDI pitch number), and encoded in grid representation with sixteenth note resolution (right). The soprano and alto voices have quarter notes at pitches G4 (67) and D4 (62), the tenor has eighth notes at pitches B3 (59) and A3 (57), and the bass has eighth notes at pitches A2 (45) and G2 (43).

MIDI-like event-based (used for the Maestro dataset)

The second dataset, Maestro (Hawthorne et al., 2019), consists of polyphonic piano performances with expressive timing and dynamics. The time resolution here is on the millisecond level, so a grid representation would result in sequences that are too long. Instead, the polyphonic performance is serialized into a sequence of one hot encoded events as proposed in Oore et al. (2018).

First, the input MIDI files are preprocessed to extend note durations based on sustain pedal control events. The sustain pedal is considered to be down whenever a sustain control change is encountered with a value $>= 64$; the sustain pedal is then considered up after a control change with a value < 64. Within a period where the sustain pedal is down, the duration of each note is extended to either the beginning of the next note of the same pitch or the end of the sustain period, whichever happens first. If the original duration extends beyond the time when the sustain pedal is down, that original duration is used.

Next, the MIDI note events are converted into a sequence from the following set of vocabulary: 128 `NOTE_ON` events for starting a note of with one of the 128 MIDI pitches, 128 `NOTE_OFF` events for ending a note with one of the 128 MIDI pitches, 100 `TIME_SHIFT` events representing forward time shifts in 10ms increments from 10ms to 1s, and 32 `SET_VELOCITY` events representing the velocity for future `NOTE_ON` events in the form of the 128 possible MIDI velocities quantized into 32 bins. An example performance encoding is illustrated in Figure 5.4.



```
SET_VELOCITY<80>, NOTE_ON<60>
TIME_SHIFT<500>, NOTE_ON<64>
TIME_SHIFT<500>, NOTE_ON<67>
TIME_SHIFT<1000>, NOTE_OFF<60>, NOTE_OFF<64>, NOTE_OFF<67>
TIME_SHIFT<500>, SET_VELOCITY<100>, NOTE_ON<65>
TIME_SHIFT<500>, NOTE_OFF<65>
```

Figure 5.4: A snippet of a piano performance visualized as a pianoroll (left) and encoded as performance events (right, serialized from left to right and then down the rows). A C Major chord is arpeggiated with the sustain pedal active. At the 2-second mark, the pedal is released, ending all of the notes. At the 3-second mark, an F is played for .5 seconds. The C chord is played at velocity 80 and the F is played at velocity 100.

■ 5.4.2 Results on the J.S. Bach Chorales dataset

The J.S. Bach Chorales dataset has been used for evaluating generative models for music [3] (e.g., Allan and Williams, 2005; Boulanger-Lewandowski et al., 2012; Liang, 2016; Hadjeres et al., 2016; Huang et al., 2017). It consists of score-based four-part chorales. We first discretize the scores onto a 16th-note grid, and then serialize them by iterating through all the voices within a time step and then advancing time (see 5.4.1 for more details). As there is a direct

[3] JSB Chorales dataset: `https://github.com/czhuang/JSB-Chorales-dataset`

correspondence between position in sequence and position on the timing/instrument grid in a piece, adding relative position representations could make it easier to learn this grammar. We indeed see relative attention drastically improve negative log-likelihood (NLL) over baseline Transformer (Table 5.2). This improvement is also reflected in sample quality. The samples now maintain the necessary timing/instrument grid, always advancing four steps before advancing in time. As local timing is maintained, the model is able to capture timing on a more global level, giving rise to regular phrasing, as shown in the right score of Figure 5.5.

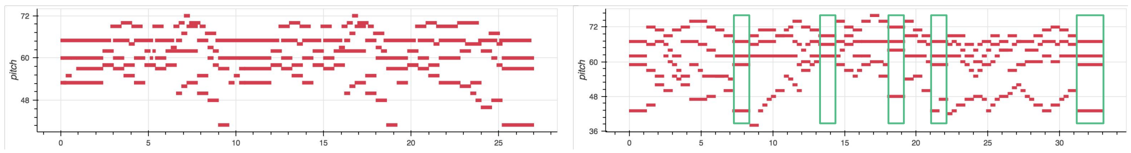


Figure 5.5: Unconditioned samples from Transformer without (left) and with (right) relative self-attention. Green vertical boxes indicate the endings of (sub)phrases where cadences are held.

In addition to relative attention, we explored enhancing absolute timing through concatenating instead of adding the sinusoids to the input embeddings. This allows the model to more directly learn its absolute positional mapping. This further improves performance for both the baseline and relative transformer (Table 5.2). We compare against Coconet as it is one of the best-performing models that has also been evaluated on the 16-note grid using the canonical dataset split. To directly compare, we re-evaluated Coconet to obtain note-wise losses on the validation set [4]. For the Transformer models (abbreviated as TF), we implemented our attention mechanisms in the Tensor2Tensor framework (Vaswani et al., 2018). We use 8 heads, and keep the query, key (att) and value hidden size (hs) fixed within a config. We tuned number of layers (L in {4,5,6}), attention hidden size (att in {256, 512}) and pointwise feedforward hidden size (ff in {512, 1024}).

Generalizing relative attention to capture relational information

A musical event bears multiple attributes, such as timing, pitch, instrument etc. To capture more relational information, we extend relative attention to capture pairwise distances on additional attributes. We learn separate relative embeddings for timing E^t and also pitch E^p. E^t has entries corresponding to how many sixteenth notes apart are two positions in time, while E^p embeds the pairwise pitch interval. However this approach is not directly scalable

[4]Some earlier papers report frame-wise losses to compare to models such as RNN-RBM which model "chords". Coconet can be evaluated under note-wise or frame-wise losses.

beyond J.S. Bach Chorales because it involves explicitly gathering relative embeddings for R^t and R^p, resulting in a memory complexity of $O(L^2D)$ as in Shaw et al. (2018). This is due to relative information being computed based on content as opposed to content-invariant information such as position in sequence. It was sufficient to add these extra relational information to the first layer, perhaps because it is closest to the raw input content. Here, the relative logits are computed from three terms, $S^{rel} = \text{Skew}(QE^r) + Q(R^t + R^p)$ in contrast with other layers that only have one relative term, $S^{rel} = \text{Skew}(QE^r)$.

5.4.3 Results on the Maestro dataset

MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization) (Hawthorne et al., 2019) is a dataset consisting of 172 hours of virtuosic piano performances captured in both MIDI and audio format [5]. It is curated from the Piano-e-Competitions [6], and proposes a train / valid / test split where the same composition, even if performed by multiple contestants, only appears in one split. The result is 295 / 60 / 75 unique compositions, corresponding to 954 / 105 / 125 performances that last for 140 / 15 / 17 hours and contain 5.06 / 0.54 / 0.57 million notes. The polyphonic MIDI performances contain both expressive dynamics and timing and we serialize them into sequences of event-based tokens as introduced in Oore et al. (2018) (see Section 5.4.1 for more details on the encoding procedure).

We train on random crops of 2048-token sequences and employ two kinds of data augmentation: pitch transpositions uniformly sampled from $\{-3, -2, \ldots, 2, 3\}$ half-steps, and time stretches uniformly sampled from the set $\{0.95, 0.975, 1.0, 1.025, 1.05\}$. For evaluation, we segment each sequence sequentially into 2048 length subsequences and also keep the last subsequences that are of shorter lengths. This results in 1128 and 1183 subsequences in the validation and test set respectively. Each subsequence is then evaluated by running the model forward once with teaching forcing. As the subsequences vary in length, the overall negative loglikelihood (NLL) is averaged entirely on the token level.

We compare our results to PerformanceRNN (LSTM, which first used this dataset) (Oore et al., 2018) and LookBack RNN (LSTM with attention) (Waite, 2016). LookBack RNN uses an input representation that requires monophonic music with barlines which is information

[5] Maestro dataset: urlhttps://magenta.tensorflow.org/datasets/maestro

[6] Piano-e-Competition: `http://www.piano-e-competition.com/`

[7] Coconet is an instance of OrderlessNADE, which approximates a mixture model over orderings where orderings are assumed to be uniformly distributed. Hence, its loss is computed by averaging losses over multiple random orderings. The current row in the table reports this loss. In contrast, the row above corresponds to evaluating Coconet as an autoregressive model under the chronological ordering. Huang et al. (2017) show that sample quality is better when using Gibbs sampling (which uses conditionals from multiple orderings) as opposed to autoregressive generation (which only uses conditionals from one ordering).

Table 5.2: Note-wise validation NLL (nats/token) on J.S. Bach Chorales where each token is a 16th note. NLL is improved by using the Transformer with our memory-efficient relative global attention formulation, also when including additional positional and relational information.

Model variation	Validation
COCONET (CNN, chronological, 64L, 128 3x3f)	0.436
COCONET (CNN, orderless, 64L, 128 3x3f)	0.238 [7]
Transformer (TF) baseline (5L, 256hs, 256att, 1024ff, 8h)	0.417
+ concat positional sinusoids	0.398
+ concat positional sinusoids, include instrument labels	0.370
TF with efficient relative attention (ours) (5L, 512hs, 512att, 512ff, 256r, 8h)	0.357
+ concat positional sinusoids, include instrument labels	0.347
+ relative pitch and time	0.335

Table 5.3: Validation NLL (nats/token) for Maestro dataset, with event-based representation with lengths $L = 2048$. Training and/or evaluating on different lengths will result in different losses because the amount of context available to the model would be different. The Transformer with our memory-efficient relative attention formulation achieves state-of-the-art results.

Model variation	Validation	Test
PERFORMANCE RNN (LSTM) (3L, 1024hs)	2.094	—
LSTM with attention (3L, 1024hs, 1024att)	2.066	—
Transformer (TF) baseline (8L, 384hs, 512att, 1024fs, 0.2d)	1.835	1.813
local attention (10L, 512bs, 0.3d)	1.895	1.888
efficient relative local attention (ours) (8L, 512bs, 0.2d)	1.873	1.861
efficient relative global attention (ours) (8L, 1024r, 0.2d)	1.808	**1.791**

that is not present in performed polyphonic music data, hence we simply adopt their architecture. Table 5.3 shows that Transformer-based architectures fits this dataset better than LSTM-based models.

We implemented our attention mechanisms in the Tensor2Tensor framework (Vaswani et al., 2018), and used the default hyperparameters for training, with 0.1 learning rate and early stopping. We compare four architectures, varying on two axes: global versus local, and regular versus relative attention. We found that reducing the query and key channel size (att) to three forth of the hidden size (hs) works well for this dataset and used this setup for all of the models, while tuning on number of layers (L) and dropout rate (d). We use block size (bs) 512 for local attention (Liu et al., 2018; Parmar et al., 2018). For relative global attention, the maximum relative distance to consider is set to half the training sequence length. For relative local attention, it is set to the full memory length which is two blocks long.

Table 5.3 shows that our memory-efficient relative attention formulations outperform regular attention in both the global and the local case. When looking at the other axes, we see global attention outperforming local attention in both the relative and regular case. Global attention may have the advantage of being able to directly look back for repeating motifs. With a larger dataset, local attention may fare well as it allows for much deeper models and longer sequences, as seen in text and image generation work (Liu et al., 2018; Parmar et al., 2018)). In turn, both domains could benefit from the use of relative local attention.

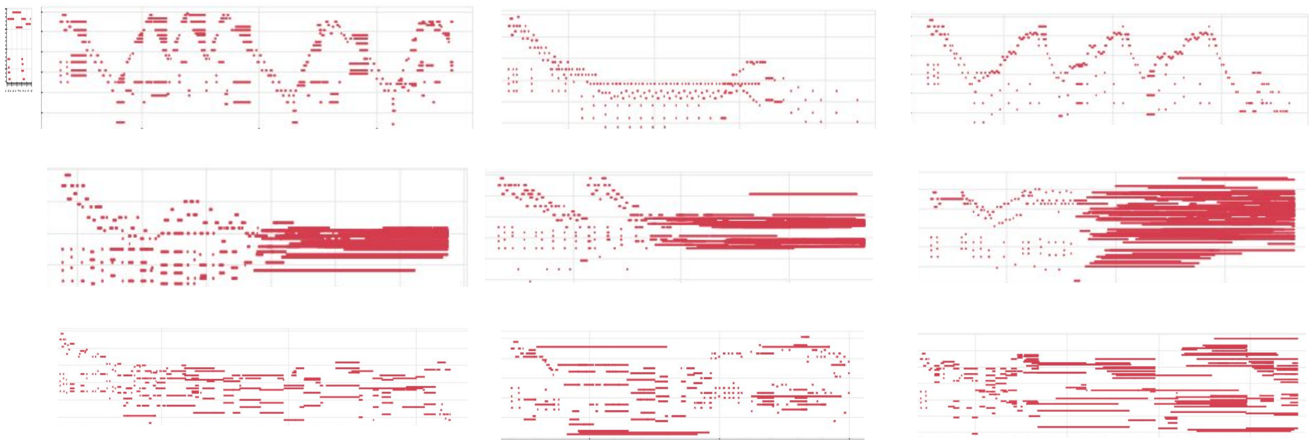


Figure 5.6: Transformer with relative attention (three random samples on the top row) when given an initial motif (top left) generates continuations with more repeated motifs and structure then baseline Transformer (middle row) and PerformanceRNN (bottom row).

Qualitative priming experiments

When primed with an initial motif (Chopin's Étude Op. 10, No. 5) shown in the top left corner of Figure 5.6, we see the models perform qualitatively differently. Transformer with relative attention elaborates the motif and creates phrases with clear contour which are repeated and varied. Baseline Transformer uses the motif in a more uniform fashion, while LSTM uses the motif initially but soon drifts off to other material. Note that the generated samples are twice as long as the training sequences. Relative attention was able to generalize to lengths longer than trained but baseline Transformer deteriorates beyond its training length.

Harmonization: Conditioning on melody

To explore the sequence-to-sequence setup of Transformers, we experimented with a conditioned generation task where the encoder takes in a given melody and the decoder has to realize the entire performance, i.e. melody plus accompaniment. The melody is encoded as a sequence of tokens as in Waite (2016), quantized to a 100ms grid, while the decoder uses the performance encoding described in Section 5.3.1 (and further illustrated in 5.4.1). We use relative attention on the decoder side and show in Table 5.4 that it also improves performance.

Table 5.4: Relative attention improves conditioned negative logliklihood (NLL) given groundtruth melodies from the validation split of the Maestro dataset.

Model variation	Validation NLL
Transformer baseline	2.066
Transformer with efficient relative attention (ours)	1.786

5.4.4 Human evaluations

To compare the perceived sample quality of models trained on the Maestro dataset, and their ability to generate a continuation for a priming sequence, we carried out a listening test study comparing the baseline Transformer, our Transformer with relative-attention, PerformanceRNN (LSTM), and the validation set. Participants were presented with two musical excerpts (from two different models that were given the same priming sequence) and asked to rate which one is more musical on a Likert scale. For each model, we generated 10 samples each with a different prime, and compared them to three other models, resulting in 60 pairwise comparisons. Each pair was rated by 3 different participants, yielding a total of 180 comparisons.

Figure 5.7 shows the number of comparisons in which an excerpt from each model was selected as more musical. The improvement in sample quality from using relative attention over the baseline Transformer model was statistically significant (see Section 5.4.4 for the analysis), both in aggregate and between the pair. Even though in aggregate LSTMs performed better in the study than the Transformer, despite having higher perplexity, but when compared against each other head to head, the results were not statistically significant (see Table 5.5).

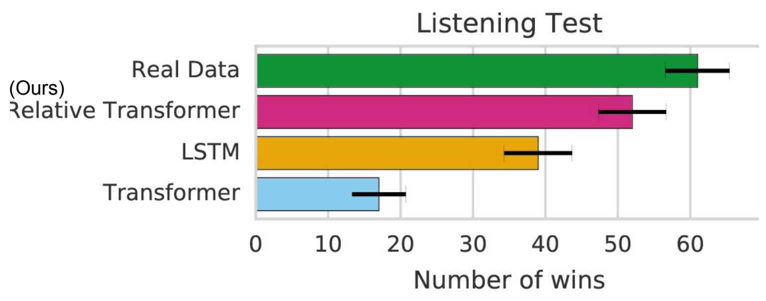


Figure 5.7: Samples from the Transformer with our efficient relative attention were rated as more musical more times than LSTM and baseline Transformer. Error bars show standard deviation of the mean.

More detailed analysis on the listening test results

A Kruskal-Wallis H test of the ratings showed that there was a statistically significant difference between the models: $\chi^2(2) = 63.84, p = 8.86\text{e-}14 < 0.01$. Table 5.5 shows the post-hoc analysis on the comparisons within each pair, using the Wilcoxon signed-rank test for matched samples. Table 5.6 shows a post-hoc analysis of how well each model performed when compared to all pairs, and compares each model's aggregate against each other, using the Mann–Whitney U test for independent samples. We use a Bonferroni correction on both to correct for multiple comparisons. The win and loss counts bucket scores are 4 and 5, and scores 1 and 2 respectively, while the tying score is 3.

Both within pairs and between aggregates, participants rated samples from our relative Transformer as more musical than the baseline Transformer with $p < 0.01/6$. For within pairs, we did not observe a consistent statistically significant difference between the other model pairs, baseline transformer versus LSTM and LSTM versus relative Transformer.

When comparing between aggregates, LSTM was overall perceived as more musical than baseline Transformer. Relative Transformer came a bit close to outperforming LSTM with $p = 0.018$. When we listen to the samples from the two, they do sound qualitatively different. Relative Transformer often exhibits much more structure (as shown in Figure 5.6), but the effects were probably less pronounced in the listening test because we used samples around 10s to 15s, which is half the length of those shown in Figure 5.6 to prevent the baseline Transformer from deteriorating. This weakens the comparison on long-term structure.

When compared to real music from the validation set, we see that in aggregates, real music was better than LSTM and baseline Transformer. There was no statistical significant difference between real music and relative Transformer. This is probably again due to the samples being too short as real music is definitely still better.

Table 5.5: A post-hoc comparison of each pair on their pairwise comparisons with each other, using the Wilcoxon signed-rank test for matched samples. p value less than 0.01/6=0.0016 yields a statistically significant difference and is marked by asterisk.

Pairs		wins	ties	losses	p value
Our relative transformer	real music	11	4	15	0.243
Our relative transformer	Baseline transformer	23	1	6	0.0006*
Our relative transformer	LSTM	18	1	11	0.204
Baseline transformer	LSTM	5	3	22	0.006
Baseline transformer	real music	6	0	24	0.0004*
LSTM	real music	6	2	22	0.0014

Table 5.6: Comparing each pair on their aggregates (comparisons with all models) in (wins, ties, losses), using the Mann–Whitney U test for independent samples.

Model		Model		p value
Our relative transformer	(52, 6, 32)	real music	(61, 6, 23)	0.020
Our relative transformer	(52, 6, 32)	Baseline transformer	(17, 4, 69)	1.26e-9*
Our relative transformer	(52, 6, 32)	LSTM	(39, 6, 45)	0.018
Baseline transformer	(17, 4, 69)	LSTM	(39, 6, 45)	3.70e-5*
Baseline transformer	(17, 4, 69)	real music	(61, 6, 23)	6.73e-14*
LSTM	(39, 6, 45)	real music	(61, 6, 23)	4.06e-5*

5.5 Conclusion

In this work we demonstrated that the Transformer equipped with relative attention is very well-suited for generative modeling of symbolic music. The compelling long-term structure in the samples from our model leaves us enthusiastic about this direction of research. Moreover, the ability to expand upon a prime, in particular, suggests potential applications as creative tool.

The significant improvement from relative attention highlights a shortcoming of the original Transformer that might also limit its performance in other domains. Improving the Transformer's ability to capture periodicity at various time scales, for instance, or relations between scalar features akin to pitch could improve time-series models. Our memory-efficient implementation enables the application of relative attention to much longer sequences such as long texts or even audio waveforms, which significantly broadens the range of problems to which it could be applied.

Chapter 6

Conclusion

This thesis shows how we can decompose support in the creative process, presenting three different points of entry for machine learning models, through generation, recommendation and control. These models can potentially extend our creative imagination by providing a wider range of possibilities, both in raw material and also in variations and extensions to our own ideas. Models that allow for flexible conditioning such as Coconet can potentially support interactive composing, allowing us to engage in a creative dialogue with generative models. Recommendation systems designed for the creative context can highlight adventurous yet relevant choices, making it easier for us step outside of our comfort zones. Personalized controls allow us to communicate in a more intuitive fashion, making it easier to be creative.

6.1 Contributions and creative implications

Music Transformer (Huang et al., 2019) advanced what we thought was possible with generative models of music, generating human-like performed music with minute-long coherence. To achieve this, we proposed a memory efficient formulation of relative attention, allowing self-attention to capture relational features while still scale to modeling few thousands of tokens. The model's ability to keep track of longer term context and to reuse and build on previous passages makes it a promising building block for meaningful musical exchange with musicians.

Coconet (Huang et al., 2017) generalizes how we factorize the musical score, and shows how we can train a model that can perform a wide range of musical tasks, from unconditioned generation, to melody harmonization to partial score competition. It uses Gibbs sampling to rewrite its own outputs to improve sample quality. This opens up the possibility of collaborative rewriting with humans.

ChordRipple (Huang et al., 2016) demonstrates a generative recommendation system that anticipates the rewrites necessary by directly recommending them along with the initial suggested changes. By using chord embeddings learned from CHORD2VEC, CHORDRIPPLE is able to surface more adventurous yet relevant chords in its recommendations. User studies show with the tool novice musicians were able to generate more novel chord progressions. This suggests that recommendations can be a very productive way to support creativity.

Active learning of high-level control knobs (Huang et al., 2014) allows users to define their own control knobs and then directly control that attribute during sound synthesis. This approach allows users to express their ideas on a much higher level of abstraction, by-passing the need to manually work out the details at every level.

■ 6.2 Recent follow-up work and discussion

■ 6.2.1 Modeling long-term structure

A key challenge in generative modeling is capturing long-term structure. These dependencies manifest themselves in different forms across domains, from surface repetitions of motifs in music and paintings or verse and chorus in lyrics, to latent context of different levels of abstraction that persist and evolve such as key in a musical piece, topic in a dialogue, and mood in various forms of narrative.

Visualizing long-term dependencies MUSIC TRANSFORMER (hua) significantly improved the long-term coherence in music samples generated by neural networks, showing it can reuse and develop motifs over thousands of tokens and as a result construct a dynamic arc across multiple phrases. These results suggest self-attention could be a good match for music. Music draws heavily on repetition and it is possible that self-attention could capture these by acting as a kind of differentiable self-similarity. Preliminary work in visualizing self-attention structure show that the weights are higher for the memory entries that are similar to the query motif (Huang et al., 2018b). Figure 6.1 shows an example of how attentions looks back at previous instances of similar motives when repeating them [1].

Relational features and generalization MUSIC TRANSFORMER uses relative attention to better capture timing, distance-dependent and periodic features. Relative attention and its variations have been shown to improve performance on language-based tasks such as translation (Shaw et al., 2018) and later language modeling (Dai et al., 2019). We proposed a memory efficient formulation, allowing it to scale to longer sequences. This formulation has already improved image generation in follow-up work to Image Transformer (Parmar et al., 2018), and can

[1] A playback animation of the self-attention visualization can be seen at
`https://magenta.tensorflow.org/music-transformer`

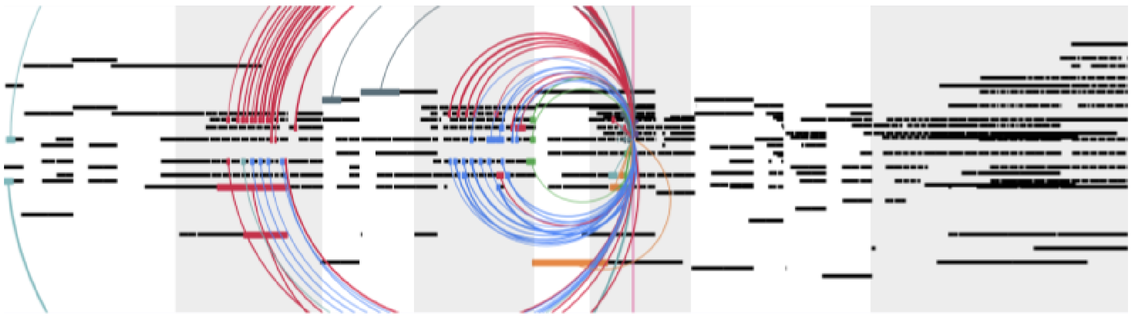


Figure 6.1: MUSIC TRANSFORMER generates music from scratch with repeated motives (manually marked by grayed out blocks). The colored arcs visualize multi-head self-attention weights and show how previous instances of motifs inform future occurrences, and how the model is able to skip over sections that are less relevant.

potentially be used for tasks that require longer context such as document generation. Moreover, the ability for relative attention to generalize beyond the length it is trained on also makes it potentially very suitable for dialogue generation where variable lengths are common.

Discussion: Only occasionally generates large-scale form The generated samples from MUSIC TRANSFORMER often do not yet exhibit structure on the level of forms such as ABA, analogous to Verse Chorus Verse song forms in popular music. This suggests the model finds it difficult to decide when to break away from the initial A section, to compose a contrasting B section that carries a significant departure, to know when to return to the earlier A section and to remember it well enough to predict it again. Instead, it sounds like the model is improvising, skillfully stringing together many plausible moments. The resulting samples are still stylistically coherent and often musically interesting, but not what we would expect given that the training data was highly structured composed music. Similar challenges exist when modeling dialogue, where a chatbot drifts from topic to topic. This may point to the need for hierarchical models that first generate a sequence of latents and then elaborate to produce the details. This approach has been used in dialogue (Serban et al., 2016), music (Roberts et al., 2017), and video (Denton and Fergus, 2018). Discrete latents were introduced to add a discrete bottleneck to continuous encodings (van den Oord et al., 2017), and have been used in the context of fast decoding for translation (Kaiser et al., 2018), generating coherent waveforms on the level of 10s (Dieleman et al., 2018), among others.

Enabling Wave2Midi2Wave: generating coherent waveforms on the level of 100s To generate coherent music on the waveform level, another approach is to factorize the generative process into two stages, first a music language model (such as MUSIC TRANSFORMER generates the notes, and then a audio synthesis model (MIDI2WAVE) turns the notes into waveforms. The first stage requires high-quality encodings of music in symbolic form, such as in the MIDI protocol, and the second stage requires aligned sequences of notes and high-quality audio are needed. However, both kinds of data are rare. Hawthorne et al. (2019) introduces the Maestro dataset from the Piano E-competition, where pianists performed on Yamaha Disklaivers, resulting in both a MIDI and audio recording of their performance. The paper then proposes to first use the aligned dataset to train a transcription model (WAVE2MIDI) that goes from audio to notes, and then uses this model to transcribe large amounts of audio. The transcribed notes in MIDI can then be used to train expressive language models. In particular, MUSIC TRANSFORMERS was used as it is capable of generating coherent composition and performance on the scale of minutes. The generated symbolic music can then be used as conditioning signal for a WaveNet (van den Oord et al.) trained for audio synthesis. Hence, the overall approach, called Wave2midi2wave consists of training a suite of models capable of transcribing, composing and synthesizing audio waveforms to generate coherent musical structure on timescales ranging six orders of magnitude (∼0.1 ms to ∼100s).

6.2.2 Towards mixed-initiative music composition

Beyond autoregressive generation Generating from left to right limits the kinds of human-in-the-loop interactions possible. COCONET shows that by modeling all ways of traversing a score, we obtain a verstile model that can complete arbitrary incomplete scores. This modeling approach has been extended to the Music Transformer, to support infilling, that is filling in the middle given the left and right context (Ippolito et al., 2018). Although not as general a factorization as Coconet, the work takes advantage of the encoder-decoder setup of the Transformer, where the encoder encodes both the left and right context, and the decoder is informed by both while filling in the blank from left to right. In language modeling, we also see that bi-directional models can provide richer feature learning than forward-only models, as a result improving downstream language understanding tasks (Devlin et al., 2018).

Bach Doodle The flexible conditioning capability of COCONET allows for the the possibility of mixed-initiative music composition, where the composer and the model can collaboratively rewrite the score. The initiative could come from the composer, where they explicitly request the model to regenerate certain parts, or the initiative could come from the model proactively propagating the necessary changes to main internal consistency of the musical score (Huang et al., 2018a).

Figure 6.2: The Bach Doodle was featured on Google's homepage from March 20th to 22nd to celebrate J.S. Bach's 334th anniversary.

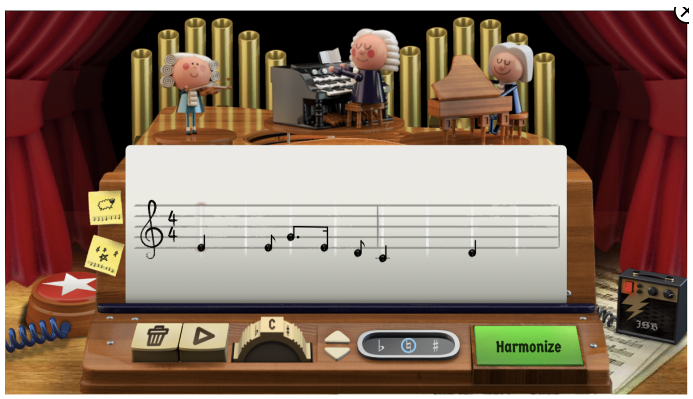


Figure 6.3: The Bach Doodle interface, where users can input a melody and then click on the green "Harmonize" button on the bottom right to request for a harmonization.

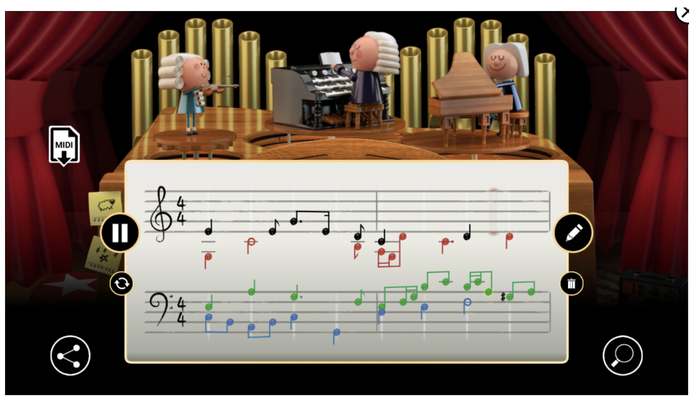


Figure 6.4: The harmonization returned by Coconet is notated in color, carrying the alto, tenor and bass voices.

Recently, COCONET powered the Bach Doodle [2] [3], harmonizing more than 50 million melodies composed by users. The Doodle was featured on Google's homepage on J.S. Bach's

[2]Bach Doodle: https://www.google.com/doodles/celebrating-johann-sebastian-bach

[3]Coconet blog post titled "The ML model behind today's Bach Doodle": https://magenta.tensorflow.org/coconet

334th Birthday, as shown in Figure 6.2. Users could input their own melody on a music-notation like interface shown in Figure 6.3, and then with a press of a button, the Doodle calls out to Coconet to harmonize the melody in the style of Bach. Figure 6.4 shows the completed score.

Machine learning can make music composition more accessible to novices. It can also make it easier for musicians, in this case to compose in a classical genre, by helping them control and curate the generation more rapidly. Artists in turn can use these models as new instruments, often in ways that we did not intend and can not foresee, by pushing them to their breaking points. Anecdotally, we observe both modes of interaction with the Bach Doodle: some chose their melodies to help the model succeed, while others experiment with how the model would react given melodies outside the Bach chorale distribution.

6.3 Future work

This thesis illustrates how machine learning can assist in the creative process through generation, recommendation and control. Music Transformer and Coconet demonstrate how we can build state-of-the-art generative models by matching the inductive bias of the models to the structure and generation process of music. Section 6.2.1 discusses approaches that could potentially capture more levels of structure.

As generative models of music continue to improve, they become musically more interesting and relevant for artists to interact with. It is interesting, then, to think about the different ways in which deep learning models may continue to be used in the future to make music composition more accessible. For these human-computer interactions to be effective, we will need to consider the range of needs that occur in composers' creative workflows.

For example, composers often express ideas on a non-musical level, such as by verbally describing how the character of a piece changes as it moves through different sections. Current music generation systems often rely on already realized music such as melody as control to constrain the kinds of output possible. To allow composers to more easily communicate their ideas with a generative model, we need to design a wider range of controls that cover describing music on different levels of abstraction.

Furthermore, musicians often need to rewrite a musical passage multiple times to find the right configuration. However, current systems are often designed implicitly for one-shot interactions. For example, they do not learn from past interactions and hence can not iteratively adapt their recommendations to user preferences.

In the following, I will discuss three ways in which future human-computer interactions could develop, given the context of these new deep learning methods.

6.3.1 Higher level controls

To control current generative models such as Music Transformer and Coconet, composers have to provide actual music or chord progressions to influence the musical output of the models. Composers often desire to sketch or fine-tune music with verbal descriptions. In the AdaptiveKnobs project, I show how sound designers can directly adjust high-level subjective quantities such as the "scariness" in sound. Active learning is used to enable users to quickly define their own high-level knobs that under the hood control multiple low-level sound synthesis parameters. This approach can also be used for symbolic music manipulation. In fact, AdaptiveKnobs was inspired by earlier work (Pachet, 2009) that uses verbal descriptions to control how an algorithm varies a given melody, for example to be more "jumpy". This is achieved by randomly varying the input melody and then ranking them according to distance from a decision boundary in a support vector machine trained to classify if a melody has a certain quality or not. The work focused on simple melodies and concepts that were closer to surface features. With models that learn rich representations on full-fledged music, we can revisit this problem and envision controls that are higher level and more relevant to musicians. Tension is an example of abstract high-level feature used by musicians. Hyperscore (Farbood, 2001) uses music-theory informed heuristics to match chord changes to tension curves. Users can then drag a tension line and have their music automatically adjusted harmonically. Morpheus (Herremans, 2016) uses a geometrical model of tonality (Chew, 2002) to infer the tension of an existing piece and uses it as a template for generating a new piece. Deep generative models could assist in uncovering more expressive features. By mapping these features to how a musician might annotate the affect of music, we could compose a piece by specifying its emotional trajectory and using it as a control signal to a generative model to realize the musical surface.

6.3.2 Adaptive recommendations

Given a musical context, whether a melody, an incomplete musical score, or a higher level conditioning signal, there is often more than one way to fill in the missing parts. Generative models give us a plausible set of solutions through its conditioned distributions. Among these, which candidate works best depends on the composer's goals and preferences. For simplicity, the Bach Doodle provides one harmonization at a time. To try out different harmonizations, the user listens to one, discards or saves, clicks the harmonize button and then listens again, and the process repeats until the user finds one that they like or they try to change up the melody to see how that impacts the results. This iteration process can be quite slow. In contrast, Chordripple (Huang et al., 2016) provides multiple recommendations at once, allowing users explore a diverse set of possibilities and potentially iterate faster.

In future work, these systems could learn from how users choose between alternatives and provide recommendations that balance exploration and exploitation. Moreover, deep generative models may offer latent dimensions that allow us to infer a composer's intention behind their surface edits. For example, if we had an embedding for chord sequences, we can observe how a sequence of edits form a trajectory in the embedding space and infer what a composer might be going for (Huang, 2016). Figure 6.5 shows a toy example of how a composer might be extending their chord progression to sound more jazzy (as indicated by the yellow arrow) and how given the direction of that change we could extrapolate (with the dotted arrow) in the embedding space to provide recommendations that anticipate further changes.

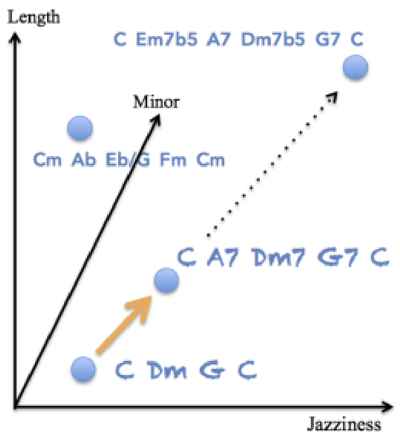


Figure 6.5: Imagined user trace (yellow arrow) and extrapolated recommendation (dotted arrow) in a chord sequence embedding space.

6.3.3 Change propagation between multiple levels of abstraction

As music consists of rich interdependencies over multiple levels within a hierarchy, it can be challenging and time consuming for composers to maintain the overall coherence of a piece when introducing edits. In ChordRipple, I proposed a ripple mechanism to automatically adjust the chords that surround the chord that was changed. However in a real composition, changing one note may require many subsequent non-local changes. Figure 6.6 (Gajos et al., 2011) shows how different levels of abstractions could be interconnected, and how a composer may have specifications for each of these levels. For example, starting with surface features on the bottom, a composer may desire a certain kind of sound that depends on how tightly a chord is voiced, on the perceived dissonance of that chord and how the dissonances of chords change across phrases to form an arc, and on the global plan for how these arcs repeat and build on each other in cycles. Changing the surface configuration such as the first note of a phrase, while maintaining these intentions, might require changing the subsequent notes in that phrase as well as the corresponding notes in other phrases in order to build a global arc. As deep generative models can learn more abstract and composite features as the depth increases, they may offer us ways to build “knobs” that allow us to control the generation of music on multiple levels of abstraction.

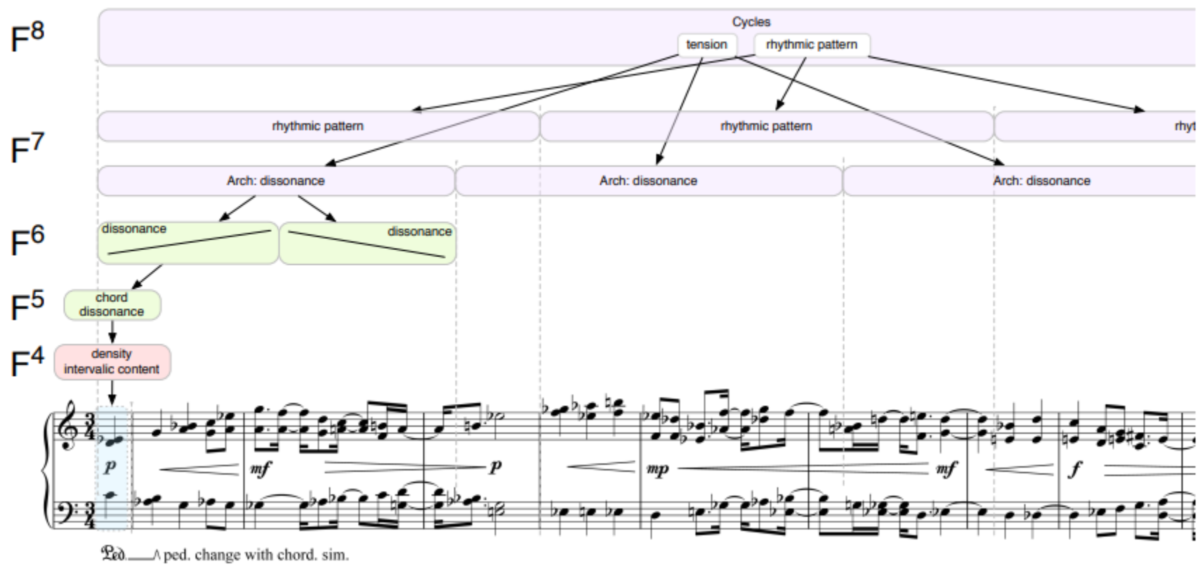


Figure 6.6: Composers have intentions on multiple levels of abstraction, which are interrelated.

Furthermore, we can build generative models that can rewrite across multiple levels, similar to how Coconet rewrites on the surface level by conditioning on one part of the score to regenerate the rest. In the future, when a composer turns a knob on one level, the model would automatically adjust the notes so that not only do it reflect this new change but also maintain the properties that were previously specified for the levels of abstraction above and below, unless the composer decides to adjust those too.

With higher level controls and adaptive recommendations, composers can explore and prototype music in a more fluid way. Imagine the best parts of jamming with another musician, and being able to do that call and response across multiple levels of abstraction.

Bibliography

M. Allan and C. K. Williams. Harmonising chorales by probabilistic inference. *Advances in neural information processing systems*, 17:25–32, 2005.

S. Amershi, J. Fogarty, A. Kapoor, and D. Tan. Effective end-user interaction with machine learning. In *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2011.

J. Bamberger. The development of intuitive musical understanding: a natural experiment. *Psychology of music*, 31(1):7–36, Jan. 2003. ISSN 0305-7356.

L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015.

N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *International Conference on Machine Learning*, 2012.

E. Brochu, T. Brochu, and N. de Freitas. A Bayesian interactive optimization approach to procedural animation design. In *Proceedings of the ACM SIGGRAPH*, 2010.

M. Cartwright and B. Pardo. Social-eq: Crowdsourcing an equalization descriptor map. In *Proceedings of the International Society for Music Information Retreival Conference*, 2013.

E. Chew. The spiral array: An algorithm for determining key boundaries. In *International Conference on Music and Artificial Intelligence*, pages 18–31. Springer, 2002.

C.-H. Chuan and E. Chew. A hybrid system for automatic generation of style-specific accompaniment. *Proceedings International Joint Workshop on Computational Creativity.*, 2007.

D. Cohn. Neural network exploration using optimal experiment design. *Neural Networks*, (6).

D. Collins. A synthesis process model of creative thinking in music composition. *Psychology of Music*, 33(2):193–216, Apr. 2005. ISSN 0305-7356.

D. Collins. Real-time tracking of the creative music composition process. *Digital Creativity*, 18(4):239–256, 2007. ISSN 14626268.

D. Collins. *The Act of Musical Composition: Studies in the creative process*. Routledge, 2016.

D. Conklin. Music generation from statistical models. In *Proceedings of the AISB Symposium on Artificial Intelligence and Creativity in the Arts and Sciences.*

D. Cope. *Computers and musical style*. Oxford University Press, 1991.

A. Cropley. In praise of convergent thinking. *Creativity Research Journal*, 18(3), 2006. URL `http://www.tandfonline.com/doi/abs/10.1207/s15326934crj1803_13`.

M. S. Cuthbert and C. Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. 2010.

Z. Dai, Z. Yang, Y. Yang, W. W. Cohen, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

F. Delalande. Towards an Analysis of Compositional Strategies. *Circuit: Musiques contemporaines*, 17(1), 2007.

E. Denton and R. Fergus. Stochastic video generation with a learned prior. *arXiv preprint arXiv:1802.07687*, 2018.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

S. Dieleman, A. van den Oord, and K. Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In *Advances in Neural Information Processing Systems*, pages 8000–8010, 2018.

H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

D. Duvenaud, J. R. Lloyd, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the International Conference on Machine Learning*, 2013.

D. Eck and J. Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, 2002.

M. Farbood. Hyperscore: A new approach to interactive computer-generated music. Master's thesis, MIT Media Laboratory, Cambridge, MA, USA, 2001.

M. Farbood and B. Schöner. Analysis and synthesis of palestrina-style counterpoint using markov chains. In *Proceedings of the International Computer Music Conference*, 2001.

J. D. Fernández and F. Vico. Ai methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48:513–582, 2013.

R. Fiebrink, P. R. Cook, and D. Trueman. Play-along mapping of musical controllers. In *Proceedings of the International Computer Music Conference*, 2009a.

R. Fiebrink, D. Trueman, and P. R. Cook. A meta-instrument for interactive, on-the-fly machine learning. In *Proceedings of the New Interfaces for Musical Expression*, 2009b.

J. Fogarty, D. Tan, A. Kapoor, and S. Winder. Cueflik: Interactive concept learning in image search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2008.

S. Fukayama, K. Yoshii, and M. Goto. Chord-sequence-factory: A chord arrangement system modifying factorized chord sequence probabilities. In *Proceedings of the International Society for Music Information Retrieval Conference ISMIR*, 2013.

J. J. Fux. *The study of counterpoint from Johann Joseph Fux's Gradus ad Parnassum*. Number 277. WW Norton & Company, 1965.

K. Z. Gajos, P. Avi, and C.-Z. A. Huang. Supporting music composition as an intentional process. In *NSF Grant Proposal*, 2011.

J. Garcia, T. Tsandilas, C. Agon, and W. E. Mackay. Structured observation with polyphony: a multifaceted tool for studying music composition. In *Proceedings of the Conference on Designing Interactive Systems*, pages 199–208. ACM, 2014.

R. A. Garcia. Automatic design of sound synthesis techniques by means of genetic programming. In *Audio Engineering Society Convention 113*, 2002.

L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

K. Goel, R. Vohra, and J. Sahoo. Polyphonic music generation by modeling temporal dependencies using a RNN-DBN. In *International Conference on Artificial Neural Networks*, 2014.

I. Goodfellow, M. Mirza, A. Courville, and Y. Bengio. Multi-prediction deep boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 548–556, 2013.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

G. Hadjeres, J. Sakellariou, and F. Pachet. Style imitation and chord invention in polyphonic music with exponential families. *arXiv preprint arXiv:1609.05152*, 2016.

G. Hadjeres, F. Pachet, and F. Nielsen. Deepbach: a steerable model for bach chorales generation. In *International Conference on Machine Learning*, pages 1362–1371, 2017.

C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck. Enabling factorized piano music modeling and generation with the maestro dataset. In *International Conference on Learning Representations*, 2019.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

S. Heise, M. Hlatky, and J. Loviscach. Automatic cloning of recorded sounds by software synthesizers. In *Audio Engineering Society Convention 127*, 2009.

D. Herremans. Morpheus: Automatic music generation with recurrent pattern constraints and tension profiles. 2016.

D. Herremans and K. Sörensen. Composing first species counterpoint with a variable neighbourhood search algorithm. *Journal of Mathematics and the Arts*, 6(4):169–189, 2012.

D. Herremans and K. Sörensen. Composing fifth species counterpoint music with a variable neighborhood search algorithm. *Expert systems with applications*, 40(16):6427–6437, 2013.

L. A. Hiller Jr and L. M. Isaacson. Musical composition with a high speed digital computer. In *Audio Engineering Society Convention 9*. Audio Engineering Society, 1957.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

M. Hoffman and P. R. Cook. Feature-based synthesis: mapping acoustic and perceptual features onto synthesis parameters. In *Proceedings of the International Computer Music Conference*, 2006.

C.-Z. A. Huang. Chordripple: Adaptively recommending and propagating chord changes for songwriters. In *Companion Publication of the 21st International Conference on Intelligent User Interfaces*, IUI, 2016.

C.-Z. A. Huang, D. Duvenaud, K. C. Arnold, B. Partridge, J. W. Oberholtzer, and K. Z. Gajos. Active learning of intuitive control knobs for synthesizers using gaussian processes. In *Proceedings of the 19th International Conference on Intelligent User Interfaces*, IUI, 2014.

C.-Z. A. Huang, D. Duvenaud, and K. Z. Gajos. Chordripple: Recommending chords to help novice composers go beyond the ordinary. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, IUI, 2016.

C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck. Counterpoint by convolution. In *Proceedings of the International Conference on Music Information Retrieval*, 2017.

C.-Z. A. Huang, S. Chen, M. Nelson, and D. Eck. Mixed-initiative generation of multi-channel sequential structures. 2018a.

C.-Z. A. Huang, M. Dinculescu, A. Vaswani, and D. Eck. Visualizing music transformer. In *NeurIPS Workshop on Interpretability and Robustness in Audio, Speech, and Language*, 2018b.

C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. Music transformer. In *International Conference on Learning Representations*, 2019.

F. Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

D. Ippolito, C.-Z. A. Huang, C. Hawthorne, and D. Eck. Infilling piano performances. In *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2018.

Ł. Kaiser, A. Roy, A. Vaswani, N. Parmar, S. Bengio, J. Uszkoreit, and N. Shazeer. Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*, 2018.

A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

A. M. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *AISTATS*, volume 1, page 2, 2011.

S. Lattner, M. Grachten, and G. Widmer. Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints. *Journal of Creative Music Systems*, 2(2), 2018.

F. Liang. Bachbot: Automatic composition in the style of bach chorales. *Masters thesis, University of Cambridge*, 2016.

J. S. Liu. The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966, 1994.

P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer. Generating wikipedia by summarizing long sequences. In *Proceedings of the International Conference on Learning Representations*, 2018.

J. Loviscach. Programming a music synthesizer through data mining. In *Proceedings of New Interfaces for Musical Expression*, 2008.

D. J. C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4, 1992.

M. Macret, P. Pasquier, and T. Smyth. Automatic calibration of modified FM synthesis to harmonic sounds using genetic algorithms. In *Proceedings of the 9th Sound and Music Computing Conference*, 2012.

R. Marsh, J. Landau, and J. Hicks. How examples may (and may not) constrain creativity. *Memory and Cognition*, 24(5):669–680, 1996.

S. Mecklenburg and J. Loviscach. subjeqt: Controlling an equalizer through subjective terms. In *CHI Extended Abstracts on Human Factors in Computing Systems*, 2006.

T. Mikolov, I. Sutskever, and K. Chen. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems.*, pages 1–9, 2013.

A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog. Retrieved June*, 20(14):5, 2015.

D. Morris, I. Simon, and S. Basu. Exposing parameters of a trained dynamic model for interactive music creation. In *Proceedings of the National Conference on Artificial intelligence*, 2008.

E. Nichols, D. Morris, and S. Basu. Data-driven exploration of musical chord sequences. *Proceedings of the 13th International Conference on Intelligent User Interfaces IUI*, page 227, 2009.

B. A. Nijstad, W. Stroebe, and H. F. Lodewijkx. Cognitive stimulation and interference in groups: Exposure effects in an idea generation task. *Journal of Experimental Social Psychology*, 38(6):535 – 544, 2002.

S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan. This time with feeling: Learning expressive musical performance. *arXiv preprint arXiv:1808.03715*, 2018.

M. A. Osborne, R. Garnett, and S. J. Roberts. Active data selection for sensor networks with faults and changepoints. *International Conference on Advanced Information Networking and Applications*, 2010.

F. Pachet. Description-based design of melodies. *Computer Music Journal*, 33(4), 2009.

F. Pachet and P. Roy. Musical harmonization with constraints: A survey. *Constraints*, 6(1): 7–19, 2001.

B. Pardo, D. Little, and D. Gergle. Building a personalized audio equalizer interface with transfer learning and active learning. In *Proceedings of the ACM workshop on Music information retrieval with user-centered and multimodal strategies*, 2012.

A. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016.

N. Parmar, A. Vaswani, J. Uszkoreit, Ł. Kaiser, N. Shazeer, and A. Ku. Image transformer. In *Proceedings of the International Conference on Machine Learning*, 2018.

D. Povey, H. Hadian, P. Ghahremani, K. Li, and S. Khudanpur. A time-restricted self-attention layer for ASR. In *Proceddings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

Z. Rafii and B. Pardo. Learning to control a reverberator using subjective perceptual descriptors. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2009.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning.* MIT Press, 2006.

A. Roberts, J. Engel, and D. Eck. Hierarchical variational autoencoders for music. In *NIPS Workshop on Machine Learning for Creativity and Design*, 2017.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

A. T. Sabin and B. Pardo. A method for rapid personalization of audio equalization parameters. In *Proceedings of the ACM International Conference on Multimedia*, 2009.

A. T. Sabin, Z. Rafii, and B. Pardo. Weighting-function-based rapid mapping of descriptors to audio processing parameters. *Journal of the Audio Engineering Society*, 59(6), 2011.

S. Seo, M. Wallat, and T. Graepel. Gaussian process regression: Active data selection and test point rejection. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2000.

I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. 2016.

J. Serrà, Á. Corral, M. Boguñá, M. Haro, and J. L. Arcos. Measuring the evolution of contemporary western popular music. *Scientific reports*, 2, 2012.

P. Shaw, J. Uszkoreit, and A. Vaswani. Self-attention with relative position representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 2, 2018.

P. Siangliulue, K. C. Arnold, K. Z. Gajos, and S. P. Dow. Toward collaborative ideation at scale: Leveraging ideas from others to generate more creative and diverse ideas. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing*, 2015.

I. Simon, D. Morris, and S. Basu. MySong: automatic accompaniment generation for vocal melodies. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, volume 1, 2008.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document, 1986.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Neural Information Processing Systems*, 2012.

R. H. Swendsen and J.-S. Wang. Nonuniversal critical dynamics in monte carlo simulations. *Physical review letters*, 58(2):86, 1987.

D. Temperley and T. D. Clercq. Statistical Analysis of Harmony and Melody in Rock Music. *Journal of New Music Research*, 42(3), 2013.

L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *In proceedings of the International Conference on Learning Representations*, 2016.

D. Tymoczko. Local harmonic grammar in western classical music. 2010.

B. Uria, I. Murray, and H. Larochelle. A deep and tractable density estimator. In *In Proceedings of the International Conference on Machine Learning*, 2014.

B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, 2016.

A. van den Oord, O. Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. N. Gomez, S. Gouws, L. Jones, L. Kaiser, N. Kalchbrenner, N. Parmar, R. Sepassi, N. Shazeer, and J. Uszkoreit. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018.

A. Venkatraman, M. Hebert, and J. A. Bagnell. Improving multi-step prediction of learned time series models. In *AAAI*, pages 3024–3030, 2015.

E. Waite. Generating long-term structure in songs and stories. https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn, 2016.

L. Yao, S. Ozair, K. Cho, and Y. Bengio. On the equivalence between deep nade and generative stochastic networks. In *In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2014.

M. J. Yee-King. An autonomous timbre matching improviser. In *Proceedings of the International Computer Music Conference*, 2011.