Corpus-based Symbolic Music Generation: Data, Representation, Models, Evaluation

by

Jeffrey Ens

B.F.A., Simon Fraser University, 2015

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Interactive Arts and Technology
Faculty of Communication, Art and Technology

Declaration of Committee

Name:	**Jeffrey Ens**
Degree:	**Doctor of Philosophy**
Thesis title:	**Corpus-based Symbolic Music Generation: Data, Representation, Models, Evaluation**
Committee:	**Chair:** Kate Hennessy Associate Professor, Interactive Arts and Technology

Ethics Statement

Abstract

Enabled by advances in artificial intelligence, research exploring the computational simulation of creative behaviours has produced human competitive generative systems in a variety of creative domains. This thesis focuses on developing generative music systems using machine learning, and evaluating these systems using statistical methods. After introducing related work in the area of generative music systems, we describe the MetaMIDI Dataset, the dataset that we will use for training, which is comprised of over 440,000 MIDI files. We adapt a pre-existing Audio-MIDI matching technique to match files in our MIDI dataset with audio previews of tracks available via the Spotify public API, since each Spotify track is associated with rich set of metadata, including information such as the artist, genre, arousal, valence and danceability. Furthermore, we provide an assessment of the accuracy of the Audio-MIDI matching technique, highlighting areas for future improvement.

Then we describe two analytic evaluation methods that we developed: CAEMSI, which is domain-agnostic; and StyleRank, which is designed for music generative systems specifically. CAEMSI is a Cross-domain Analytic Evaluation Methodology for Style-Imitation systems. Note that style-imitation systems are simply generative systems that are trained to produce artifacts in a particular style. In the context of evaluating style-imitation systems, we are often interested in determining if there is a statistically significant difference or equivalence between the training data and a set of artifacts generated by the system. To this end, we outline a statistical method to measure the equivalence and difference between two sets of artifacts, given an arbitrary similarity or distance measure. Using normalized compression distance, we conduct experiments which demonstrate that CAEMSI frequently detects a significant difference between the work of two different visual artists and detects a significant equivalence between two disjoint sets of work from the same visual artist. The same test is repeated for music composers, with similar results.

StyleRank is a system for ranking symbolic musical excerpts based on their similarity to a style. Note that we consider a style to simply be the stylistic characteristics delineated by an arbitrary collection of symbolic musical excerpts, which we refer to as the corpus. Musical excerpts are represented using a variety of features, and a Random Forest is trained to discriminate between the corpus and the set of excerpts we wish to rank. An embedding is extracted from a trained Random Forest, from which the rankings are directly derived. We outline two experiments which demonstrate that: StyleRank can proficiently distinguish between the musical styles of different composers; and that StyleRank is congruent with human perception, using data collected from thousands of participants in an online listening study. We anticipate that this system will be useful for researchers who wish to evaluate the performance of many systems, or investigate the effects of various hyper-parameters, employing experimental designs which would be incompatible with a listening test experimental design.

Motivated by lack of consensus within the research community, we make some recommendations for the listening experiment design, examining the role of two parameters, the proportion of questions and the proportion of participants, both of which are measured relative to the total number of observations. Using experimental data collected from previous studies, we compare the power and reliability of various experimental designs to arrive at substantiated recommendations regarding these proportions.

Finally, we propose the Multi-Track Music Machine (MMM), a generative system trained using the MetaMIDI dataset, that is designed to support co-creative music composition workflows. MMM supports the infilling of musical material on the track and bar level, and can condition generation on particular attributes including: instrument type, note density, polyphony level, and note duration. In order to integrate these features, we employ a different type of representation for musical material, creating a time-ordered sequence of musical events for each track and concatenating several tracks into a single sequence, rather than using a single time-ordered sequence where the musical events corresponding to different tracks are interleaved. We present experimental results which demonstrate that MMM is able to consistently avoid duplicating the musical material it was trained on, generate music that is stylistically similar (as measured using StyleRank) to the training dataset, and that attribute controls can be employed to enforce various constraints on the generated material. We also outline several real world applications of MMM, including the production of musical albums, and collaborations with industry partners that explore integrating MMM into real-world products.

Keywords: Generative Music; Evaluation; Machine Learning; Big Data

Dedication

For my loving family

Acknowledgements

First, I want to acknowledge that the work presented in this thesis has been conducted on the unceded Traditional Coast Salish Lands including the Katzie, Kwikwetlem, Kwantlen, Musqueam, Qayqayt, Semiahmoo, Squamish, Tsawwassen, Tsleil-Waututh Nations.

I would like to thank my supervisor, Philippe Pasquier, for the guidance and feedback he provided throughout my Ph.D. studies. Also I would like to thank my committe members, Steve DiPaola and Bob Sturm. Thanks to my collaborators Cale Plut and Renaud Tchemeube, as well as my colleagues in the Metacreation lab. I would also like to thank Tiffany Taylor, Lisa DaSilva and all the other SIAT administrative staff for their much appreciated assistance throughout my studies.

Thanks to my parents, who instilled in me a curiosity to learn, experiment and explore new things, and the confidence to fail sometimes.

Thanks to my daughter, who's arrival served as increased motivation to bring this chapter of my life to a close. Here's hoping that you'll be interested enough to read this someday.

Thanks to Megan. Without her support, understanding, compassion and love, I would not have been able to complete this journey. I can't thank you enough for providing an environment that is conducive to achieving my goals.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction and Motivations

1.1.1 Thesis format

This thesis takes the form of a cumulative thesis. In lieu of a monograph dissertation, cumulative theses are comprised of a collection of scholarly peer-reviewed articles. This thesis consists of a total of 5 conference papers, 4 of which have been published. Collectively, these articles summarize our contributions in two areas: developing a generative music system, and evaluation methods for generative music systems in general. The introduction outlines the research topic and factors which motivate the research questions addressed within the thesis. The second chapter summarizes related work in the area of symbolic generative music systems. The following five chapters are comprised of the aforementioned conference papers. The final chapter in the thesis summarizes the research contributions and discusses future work. Specific guidelines for a cumulative dissertation at the School of Interactive Arts and Technology can be found in Appendix A. Appendix B is a conference paper which investigates the influence of rhythm and pitch complexity on similarity judgements between melodies. Appendix C describes GenDetect, a system that was able to discriminate between generated and human-composed musical excerpts with 99% accuracy[1], winning the MIREX 2019 patterns for prediction competition. Appendix D summarizes pertinent considerations with regards to Canadian copyright law and the distribution of the MetaMIDI dataset, which is introduced in Chapter 3. Appendix E is an appendix for the StyleRank system that is introduced in Chapter 5.

1.1.2 Overview

The research presented in this thesis can be categorized into two research areas (RA): (A) the development of a generative music system; and (B) contributions related to the evaluation of generative music systems.

Research Area A : Developing Generative Music Systems

Although the music generation task has been explored in both the audio [14, 9] and symbolic domains [3], we focus on the symbolic music generation task exclusively throughout the thesis. Musical material is typically represented in two different ways: as an audio signal; or using a discrete set of symbols. In what follows, we refer to the former as *Audio* representations, and the later as *Symbolic* representations. The main aspect that distinguishes symbolic representations from audio representations, is that symbolic representations notate each discrete musical event (such as a note onset) and the time at which that event occurs. In contrast, since audio representations only provide the raw audio signal, individual events are not immediately accessible, however, computational approaches to recovering discrete information from a signal is an active area of research [17, 35, 47].

[1]https://www.music-ir.org/mirex/wiki/2019:Patterns_for_Prediction_Results

It is worth noting, that there are some inherent limitations to using symbolic representations, as it is difficult to capture complex timbres using a symbolic representation.

More specifically, this thesis explores the design and evaluation of Corpus-based Symbolic Polyphonic music Generation (CSPG) systems, which are systems that are trained to generate symbolic polyphonic music using a corpus of musical data. Although the dataset (Ch. 3) and evaluation methodologies (Ch. 4,5, and 6) we develop are equally applicable to the broader category of Symbolic Music Generation (SMG) systems, to support the narrative of the thesis, which culminates in the development and evaluation of a CSPG system (Ch. 7), we place an emphasis on CSPG systems throughout.

Research Area B : Evaluation of Generative Music Systems

In contrast to most problems that are addressed in the field of artificial intelligence, there are no optimal solutions for most creative problems, as it is simply not possible to define the optimal choreography, narrative, poetry or musical composition [41]. Although this is a subtle difference, it complicates the process of evaluating generative systems, as there is not necessarily a single correct way to conduct an evaluation. As a result, the way in which a generative system is evaluated varies, however, the evaluation methodology is typically informed by the goals of the research being conducted, and the context in which the generative system is to be used. For example, researchers with the goal of developing a system that exhibits creative behaviours in a variety of domains (i.e. general creativity), may be interested in evaluating how a generative system makes creative decisions. Alternatively, researchers with the goal of developing a generative system that can be applied in a commercial setting would likely focus on measuring attributes of the generated artifacts themselves. Note that a more detailed discussion on evaluation methodologies for CSPG systems can be found in Section 2.2.5. In this thesis, we eschew the difficulties associated with developing and evaluating a system capable of exhibiting general creativity, instead focusing on creating a usable generative system. Consequently, the evaluation methodologies that we develop and employ are focused on measuring attributes of the generated artifacts, without any consideration of the inner workings of the generative system itself.

Brief Overview of Thesis Topics

In Chapter 2, we provide an overview of CSPG systems, summarizing the datasets that have previously been employed, the manner in which musical data is symbolically represented, and the various model architectures which are employed. In Chapter 3, we introduce the Meta-MIDI dataset, a large collection of MIDI files and metadata which can be used to train CSPG/SMG systems. Chapter 4 outlines CAEMSI, which provides a domain-agnostic statistical framework for evaluating style imitation. Chapter 5 describes our work on Style Rank, a technique for ranking the similarity of musical excerpts to a style, as delineated by an arbitrary collection of musical excerpts. Chapter 6 outlines actionable recommendations for improved listening experiment design, addressing the lack of consistency in listening experiment designs. Finally, Chapter 7 outlines our own flexible CSPG system, the Multi-Track Music Machine, which we evaluate using some of the work presented in previous chapters.

1.1.3 Motivations

In this section, we discuss the broad motivating factors for our research, reserving our discussion of the motivations related to specific research questions until Section 1.2.2, where the research questions and contributions are also outlined.

Research Area A : Developing Generative Music Systems

We identify two broad motivating factors for our research involving the development of generative music systems. First of all, computational creativity explores the automation of creative processes, with the aim of deepening our understanding of human creativity through the simulation of creative behaviours, and investigating creative processes which are beyond current human capability [41]. Since creativity is a complex phenomenon, the simulation of creative behaviours is an ideal method of scientific inquiry, which has the potential to directly inform our understanding of human creativity. Increasingly, research has focused on the simulation of creative tasks via state-of-the-art techniques in artificial intelligence, developing generative systems that can produce human-competitive creative artifacts, including visual art [13], poetry [5], video game levels [49], and music [48, 43, 20].

Secondly, there is increasing demand for generative systems in creative industries, which can directly benefit from systems that automate or expedite portions of the music composition process. For example, interactive video games which allow the player to progress through the game in a non-linear manner, are an excellent match for generative music systems. The prototypical solution involving looping linear music compositions is quite rigid, and developing generative music systems that can flexibly adapt to the current game scenario is an active area of research. The demand for customized musical content extends far beyond the video game industry, as any media creator will likely need to incorporate music into the content they are producing. Due to the costs of licensing or commissioning human-composed music, there is certainly an opportunity for AI-based generative systems that can generate compelling music.

Research Area B : Evaluation of Generative Music Systems

In many lines of research, evaluation methods are a necessity, as there must be a way to quantify what is being studied. Unsurprisingly, this is the most prevalent motivation for developing and refining evaluation methodologies for generative music systems. However, in this thesis we emphasize analytic evaluation methodologies that rely purely on computation (Ch. 4 and 5), rather than on collecting feedback from human participants via a listening study, a decision which has several specific motivations.

The first motivation is related to domain knowledge. Ideally, participants in a listening study should be experienced with the style of music that is being generated by the system. Clearly, this can place limitations on the types of scientific inquiries that are addressed, as it may be difficult or expensive to source participants with sufficient domain knowledge for some types of music. Theoretically, computational approaches can obtain sufficient domain knowledge from the training corpus, which removes restrictions on the style of music contained in the training corpus.

Second, listening studies can be difficult to reproduce. In contrast to computational methods, which will produce the same result given the same input, factors such as the listening environment, cultural context [12], and expertise level [30, 31], can influence the result. Furthermore, some research [8, 39, 7], has found evidence of human bias against generative systems. Although similar studies have failed to find this same effect [37, 15], this is undoubtedly another factor which can effect the reproducibility of an experiment.

Finally, in study designs that involve human participants, there are clear limitations on the total number of observations that can be collected, as fatigue will eventually being to degrade the quality of their responses. Furthermore, the involvement of human participants imposes monetary and temporal constraints. Collectively, these factors place inherent limitations on the number of systems that can be simultaneously evaluated, and on the number of generated artifacts that can be used to evaluate each system. In contrast, computational methods are typically easily parallelizable, inexpensive and can be run non-stop until completion.

Of course, we willingly acknowledge that computational evaluation methodologies are not without their limitations. We are simply outlining the issues with listening studies that have motivated our research on computational evaluation methodologies for generative music systems.

1.2 Thesis structure

1.2.1 Research questions

In this section, we outline the research questions and summarize the motivations and contributions for each chapter of this thesis. Table 1.1 describes the papers included in the thesis and the research question(s) that they address.

RQ1 How can we build a large and rich dataset for training a CSPG/SMG system?

RQ2 How can we determine whether there is a statistically significant difference or equivalence between two sets of musical artifacts?

RQ3 How can we measure the similarity between musical artifacts?

RQ4 How can we employ statistical methods to develop evidence-based recommendations for improved listening experiment design?

RQ5 How can we create a CSPG system that accommodates flexible generation?

Chap.	RA	RQ	Contributions	Publication Reference / Publication Appendix
Ch. 1			General Introduction, Research Questions and Contributions	Appendices: B[27] & C[23]
Ch. 2			Review of CSPG systems, datasets, and evaluation methods	-
Ch. 3	A	1	A large collection of MIDI files and metadata which can be used to train CSPG/SMG systems.	Ens & Pasquier (2021) [21]
Ch. 4	B	2	A domain-agnostic statistical method to evaluate style imitation systems.	Ens & Pasquier (2018) [22]
Ch. 5	B	3	A method to rank the similarity of several MIDI files against a style delineated by a set of MIDI files.	Ens & Pasquier (2019) [26]
Ch. 6	B	4	Actionable improvements to listening experiment design that increase statistical power	Ens & Pasquier (2020) [24]
Ch. 7	A	5 & 6	The Multi-Track Music Machine, a flexible CSPG system that accommodates bar and track level inpainting, and attribute control.	-
Ch. 8			Concluding remarks and future work	-

Table 1.1: Structure of the thesis outlining Chapters (Chap.), Research Areas (RA), Research Questions (RQ), Contributions and Publications (- denotes unpublished material)

RQ6 How can we evaluate MMM?

1.2.2 Outline of Contributions and Motivations

Chapter 1 – Introduction

We introduce the material to be presented in this cumulative thesis. We present the Research Questions, and motivation for our research. We also summarize contributions, and outline the structure of the thesis.

Chapter 2 – Related Work

In Chapter 2, we provide an overview of the of CSPG systems. We discuss several factors related to CSPG systems including: the data on which the system is trained; the representation which is employed to encode musical material; the texture of music that is generated (i.e. polyphonic or monophonic); the model architecture or machine learning method that is used to learn from the data; and the evaluation methods applied to the system. Collectively these factors provide a broad lens through which we can analyze the merits of each system. We close this chapter with a discussion of the opportunities for future work, some of which are directly addressed in the remaining chapters of this thesis.

Chapter 3 – Building the MetaMIDI Dataset:Linking Symbolic and Audio Musical Data

Research questions addressed:

RQ1 How can we build a large and rich dataset for training a CSPG/SMG system?

Motivation: In Chapter 3, we describe the process of building the MetaMIDI Dataset, a collection of MIDI files linked to additional metadata. Our primary motivation for developing this dataset was twofold: first, we want to increase the amount and diversity of MIDI data available for training CSPG/SMG systems; and second, we wanted to apply audio-MIDI matching techniques to augment the MIDI data with metadata from a large audio dataset.

Contribution: We provide the MetaMIDI (MMD) dataset, a collection of 436,631 MIDI files and metadata. The MMD contains artist and title metdata for 221,504 MIDI files and genre metadata for 143,868 MIDI files, which was collected from the websites on which the MIDI files were hosted. In order to expand the metadata available for each MIDI file, we employ the same technique used to construct the Lahk MIDI Dataset [45]. Raffel et al. matched 30-second audio previews from the Million Song Dataset [1] against the MIDI files in the Lahk MIDI Dataset, to augment the metadata for each MIDI file. Since the Million Song Dataset is no longer readily available online, we elected to use the 30-second audio previews which are available through Spotify's public API. After matching the MIDI files in the MMD against 32,000,000 30-second audio clips, we produced over 10,796,557 audio-MIDI matches. In addition, we linked 600,142 Spotify tracks with 1,094,901

MusicBrainz recordings to produce a set of 168,032 MIDI files that are matched to the MusicBrainz database. To increase the reliability of the audio-MIDI matches, we also computed audio-MIDI matches where the Spotify artist and title metadata are a fuzzy match to the web-scraped metadata. This process produced 53,496 matched MIDI files. Since the Spotify API and MusicBrainz database provide extensive metadata, the links between MIDI files and these two data sources can be easily used to gather additional data about each matched MIDI file. At the time of writing, this collection of MIDI files is currently the largest available, making it a valuable asset to MIR researchers. Important considerations with regards to Canadian copyright law and the distribution of the MetaMIDI dataset are presented in Appendix D.

Chapter 4 – CAEMSI : A Cross-Domain Analytic Evaluation Methodology for Style Imitation

Research questions addressed:

RQ2 How can we determine whether there is a statistically significant difference or equivalence between two sets of musical artifacts?

Motivation: This research contribution is domain agnostic and can be applied to the evaluation of all style-imitation systems, of which CSPG/SMG systems are a subset. When evaluating a style-imitation system, which is simply a generative system trained to produce artifacts in a particular style, we are often interested in comparing the corpus ($\mathcal{C}$) and a set of artifacts generated by the system ($\mathcal{G}$). Given a suitable similarity measure, this amounts to determining if there is a statistically significant difference or equivalence between $\mathcal{C}$ and $\mathcal{G}$. Simply reporting the average similarity, does not provide sufficient information, as it does not indicate whether the observed difference or equivalence is simply a random effect. Notably, in the case of style imitation [41], we are interested in emulating the style delineated by $\mathcal{C}$, and ideally wish to demonstrate that $\mathcal{G}$ is equivalent to $\mathcal{C}$ in some respect. Since the absence of a significant statistical difference is not the same as significant statistical equivalence, it is worth developing a framework which accommodates both these types of statistical enquiries.

Contribution: We propose CAEMSI (Cross-domain Analytic Evaluation Methodology for Style Imitation systems) [22], which is based on a set of statistical tests that allow hypotheses comparing two sets of artifacts to be tested. Given a corpus $\mathcal{C} = \{\mathcal{C}^1, ..., \mathcal{C}^n\}$, and a set of generated artifacts $\mathcal{G} = \{\mathcal{G}^1, ..., \mathcal{G}^m\}$, we can compute three distributions: $S_{\mathcal{C},\mathcal{G}}$, the set of inter-set similarities between items in $\mathcal{C}$ and $\mathcal{G}$; $S_{\mathcal{C}}$, the set of intra-set similarities within $\mathcal{C}$; and $S_{\mathcal{G}}$, the set of intra-set similarities within $\mathcal{G}$. Using a permutation testing framework, we present a way to test the hypothesis that $S_{\mathcal{C},\mathcal{G}} = S_{\mathcal{C}} = S_{\mathcal{G}}$ (equivalence) and $S_{\mathcal{C},\mathcal{G}} \neq S_{\mathcal{C}} \neq S_{\mathcal{G}}$ (difference). Using the inverse of Normalized Compression Distance (NCD) as a generic similarity measure (domain specific similarity measures are preferable in practice), we measure the power of these statistical tests using the Classical Archives MIDI dataset and the WikiArt dataset. The Classical Archrives MIDI dataset consists of 14,724 compositions by 843 distinct composers. The WikiArt dataset consists of 19,052 paintings by 23 artists. We found that when comparing sets of artifacts from two different artists/composers, CAEMSI would reliably detect a significant difference, and when comparing two distinct subsets of artifacts from the same artist/composer, CAEMSI would reliably detect a significant equivalence.

Chapter 5 – Quantifying Musical Style: Ranking Symbolic Music based on Similarity to a Style

Research questions addressed:

RQ3 How can we measure the similarity between musical artifacts?

Motivation: Typically, when we are evaluating a CSPG/SMG system, we are interested in comparing two sets of musical excerpts: A set of pieces $\mathcal{C} = \{\mathcal{C}^1, ..., \mathcal{C}^n\}$ representative of the data on which the model was trained, which we refer to as the corpus; and a set of generated artifacts $\mathcal{G} = \{\mathcal{G}^1, ..., \mathcal{G}^m\}$. As a result, a reliable and meaningful measure of similarity between the corpus ($\mathcal{C}$) and a generated artifact $\mathcal{G}^i$ is needed. Although CAEMSI (Ch. 4) was able to reliably distinguish between the compositions from two different composers using a generic similarity measure (inverse of normalized compression distance), we aim to achieve improved performance using a similarity measure that is specifically designed for symbolic polyphonic music.

Contribution: We propose StyleRank [26], a method to measure the similarity between a MIDI file and an arbitrary musical style delineated by a collection of MIDI files. MIDI files are encoded using a novel set of features and an embedding is learned using Random Forests. Using this embedding, the similarity between two artifacts can be directly computed. One advantage of this approach, is that the similarity between two artifacts is informed by the characteristics of the entire set of MIDI files on which the embedding is learned. We conduct experiments which demonstrate that StyleRank is highly correlated with human perception of stylistic similarity, and that it is precise enough to rank generated samples based on their similarity to the style of a corpus. We compare StyleRank against Yang's approach [52], which involves applying classical distance metrics directly to the feature vectors, and found that StyleRank was much more accurate. Somewhat surprisingly, we found that StyleRank was more accurate than using log-likelihood to predict stylistic similarity, using data collected in the BachBot listening study [33]. Since similarity is computed in a large feature space, we can also measure similarity with respect to a single feature, allowing specific discrepancies between generated samples and a particular musical style to be identified. Building on this approach, we developed a GenDetect [23] (Appendix C) which discriminates between generated and human-composed continuations of a musical prompt. The system won the 2019 MIREX Patterns for Prediction competition, reliably discriminating generated continuations from those which were human composed over 99% of the time, on an undisclosed test set.

Chapter 6 – Improved Listening Experiment Design for Generative Systems

Research questions addressed:

RQ4 How can we employ statistical methods to develop evidence-based recommendations for improved listening experiment design?

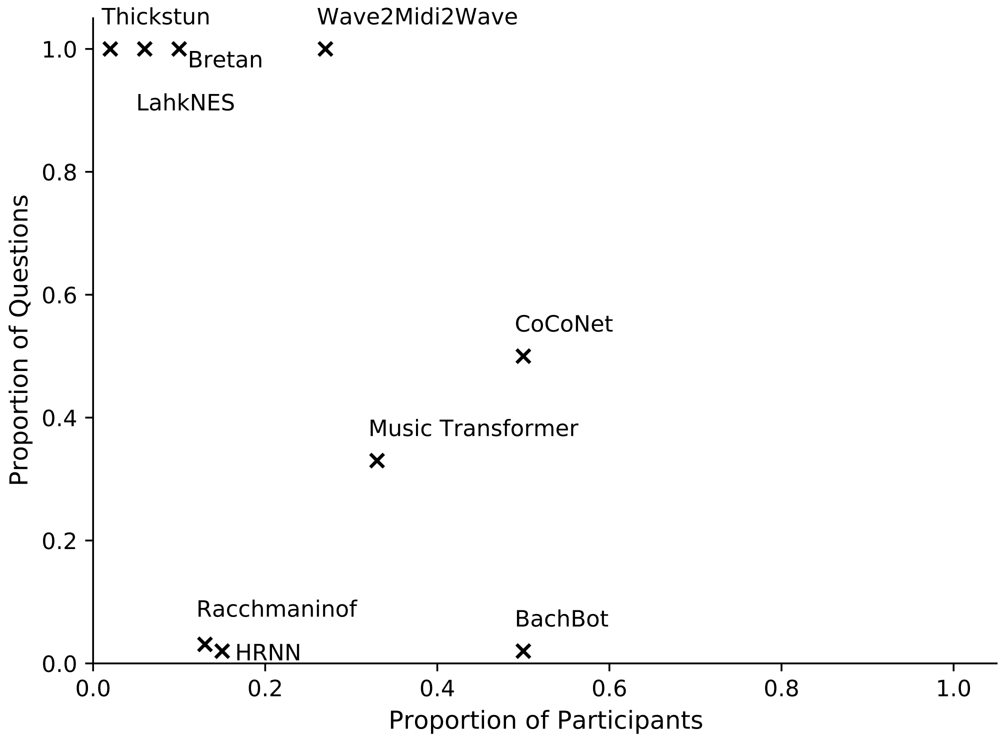


Figure 1.1: The experimental designs employed in recent listening studies for generative systems

Motivation: Despite the advantages of purely analytic methods of evaluation, many researchers still choose to conduct a listening test, which has become the standard approach for CSPG/SMG system evaluation [44]. However, there are significant discrepancies between studies. Consider an experiment $\mathcal{E} = \{(Q^{\gamma_1}, S^{\alpha_1}, R^1), ..., (Q^{\gamma_{n_{\text{obs}}}}, S^{\alpha_{n_{\text{obs}}}}, R^{n_{\text{obs}}})\}$ consisting of $n_{\texttt{obs}}$ observations (R^i), given a set of questions $Q = \{Q^1, ..., Q^{n_{\text{ques}}}\}$ and a set of participants $S = \{S^1, ..., S^{n_{\text{par}}}\}$. Note that a question is simply a set of musical excerpts sampled from one or more sources, from which a participant must formulate a response (R^i). Given $\mathcal{E}$, the proportion of participants is $n_{\texttt{par}}/n_{\texttt{obs}}$ and the proportion of questions is $n_{\texttt{ques}}/n_{\texttt{obs}}$. In Figure 1.1, we plot the proportion of questions $(\frac{n_{\text{ques}}}{n_{\text{obs}}})$, the proportion of participants $(\frac{n_{\text{par}}}{n_{\text{obs}}})$, the experimental design, and the methodology for several recent listening experiments for which the relevant information was available. The noticeable discrepancies with respect to the proportion of questions and proportion of participants, indicates a lack of consensus amongst the research community, directly motivating a rigorous analysis of these experimental hyper-parameters.

Contribution To answer this research question, we examine the role of two parameters, the proportion of questions ($\frac{n_{\texttt{ques}}}{n_{\texttt{obs}}}$) and the proportion of participants ($\frac{n_{\texttt{par}}}{n_{\texttt{obs}}}$), through the lens of two experiments [24]. In the first experiment, we conduct a parameter sweep, manipulating the proportion of participants and questions, using power calculations designed for experiments with two random factors [51]. The results demonstrate that increasing the proportion of questions and participants results in increases in experimental power. Notably, we observe limitations to the amount of power that can be gained by manipulating a single parameter (either the proportion of participants or questions). The second experiment attempts to quantify the inter-experiment variance by simulating thousands of miniature experiments, using the raw experimental data collected from previous listening studies. The results in the second experiment corroborate findings in the first experiment, demonstrating that inter-experimental variance is minimized when the proportion of questions and participants are as close to 1 as possible. Fortunately, these issues can be fairly easily addressed, as increasing the number of questions simply involves collecting more samples from each generative system being studied.

Chapter 7 – MMM : Exploring Conditional Multi-Track Music Generation with the Transformer

Research questions addressed:

RQ5 How can we create a CSPG system that accommodates flexible generation?

RQ6 How can we evaluate MMM?

Motivation: Although one-shot generation of musical material is impressive from a technical standpoint, it is not that useful in a practical context, as the composition process is frequently an iterative process which involves gradually refining a piece of music. As a result, there is a large incentive to develop models which accommodate the natural workflow of the composer, so these systems can be adopted into the music making process. As a result, we aim to develop a CSPG system that generates multi-track musical material, offering control over musical attributes, as well as the flexibility of track-level and bar-level inpainting. Although there has been some work in this area [19, 32, 43, 36] (Table 1.2), at the time of writing, there are no systems which accommodate both inpainting and attribute control.

Contribution: We developed the Multi-Track Music Machine (MMM) [25], a generative system based on the Transformer architecture that is capable of generating multi-track music. In contrast to previous work, which represents musical material as a single time-ordered sequence, where the

	System	Model	Dataset	Continuation	Note-Level Inpainting	Bar-Level Inpainting	Track-Level Inpainting	Attribute Control	Num Mono. Tracks	Num Poly. Tracks	Num Instruments
C1	CoCoNet[19]	CNN	Bach Chorales (DT2)	×	×	×	×		4	0	4
C2	C-RBM[32]	CNN	Mozart/Batik (DT9)	×	×	×			-	1	1
D1	Hild[18]	FFNN	Bach Chorales (DT2)	×					4	0	4
E1	MuseGan[11]	GAN	Lakh Midi (DT7)	×			×		-	4	4
E2	C-RNN-GAN[38]	GAN	Classical (3697) ⋆	×					-	1	1
F1	DeepBach[16]	RNN	Bach Chorales (DT2)	×			×		4	0	4
F2	Josyln et al.[29]	RNN	Nottingham (DT4)	×					1	1	2
F3	DeepJ[34]	RNN	Classical ⋆	×			×		-	1	1
F4	Johnson[28]	RNN	Bach Chorales (DT2) Nottingham (DT4) Musedata (DT5) Piano-Midi.de (DT6)	×					-	1	1
F7	RNN-RBM[2]	RNN	Bach Chorales (DT2) Nottingham (DT4) Musedata (DT5) Piano-Midi.de (DT6)	×					-	1	1
F8	BachBot[33]	RNN	Bach Chorales (DT2)	×			×		4	0	4
F9	Choi et al.[6]	RNN	Jazz Lead Sheets ⋆	×					-	1	1
F10	Meade et al.[36]	RNN	Piano-e- Competition (DT3)	×				×	-	1	1
F11	PerformanceRNN[40]	RNN	Piano-e- Competition (DT3)	×					-	1	1
F12	Walder et al.[50]	RNN	Bach Chorales (DT2) Nottingham (DT4) Musedata (DT5) Piano-Midi.de (DT6)	×					-	1	1
-	FolkRNN [48]	RNN	23,000 Folk Transcriptions	×					-	-	-
G1	MuseNet[43]	Trans.	Custom ⋆	×				×	-	10	10
G2	Music Trans.[20]	Trans.	Maestro Dataset (DT10)	×					-	1	1
G5	LahkNES[10]	Trans.	Lahk Midi (DT7) NESMDB (DT13)	×					4	0	4
H1	MidiVAE[4]	VAE	Jazz / Pop / Classical ⋆	×					-	1	1
H2	MusicVAE[46]	VAE	Custom (1.5 million) ⋆	×				×	3	0	3
-	InpaintNet[42]	VAE	-	×		×			1	0	1
G3	**MMM[25]**	Trans.	MetaMIDI (Ch. 3)	×		×	×	×	-	12	128

Table 1.2: A summary of relevant CSPG/SMG systems. An × is used to denote systems that are capable of a particular sampling method.

musical events corresponding to different tracks are interleaved, we create a time-ordered sequence of musical events for each track and concatenate several tracks into a single sequence. This takes advantage of the Transformer's attention-mechanism, which can adeptly handle long-term dependencies. We explore how our novel representations can offer the user a high degree of control at generation time, accommodating track-level and bar-level inpainting while offering control over track instrumentation and per-track note density.

In contrast to previous systems (see Table 1.2), which support at most 10 different instruments [43], MMM supports each of the 128 different general MIDI instruments. Furthermore, MMM allows for an arbitrary collection of tracks to be used. This differs from all of the systems in Table 1.2, which are trained on a fixed set of tracks. For example, MusicVAE [46] can only handle drum, bass and melody trios while LahkNES [10] only accommodates 4 track NES style arrangements. Although MuseNet offers the user control over the instruments to be used in the generated material, the model only treats this input as a suggestion, and may generate a subset or super-set of the suggested instruments. In contrast, instrument selection with MMM is guaranteed. One current limitation of MMM, is that the length of musical material that can be generated is limited by the size of the attention window, placing an upper bound on the number of bars that can be generated.

Notably, there have been several real-word applications of MMM, including integration into music software, collaborations with industry partners, and most importantly, music that was produced using the system. More detailed information on these various applications can be found elsewhere[2].

Chapter 8 – Conclusion

We conclude this thesis by summarizing and outlining the main contributions of the research, and suggest future areas for investigation.

Appendix B - The Significance of the Low Complexity Dimension in Music Similarity Judgements

Understanding how similarity is perceived by humans in a musical context is fundamental to many areas of Music Information Retrieval (MIR) research. Our study demonstrates that ones perception of musical similarity varies with respect to the musical content being compared, as the dimension (either pitch or rhythm) bearing low complexity information was found to be the predominant factor influencing similarity judgements.

Appendix C - Discriminating Symbolic Continuations with GenDetect

We propose GenDetect, an algorithm designed to discriminate between two possible continuations of a musical prompt, distinguishing human composed continuations from those which are generated.

[2]https://metacreation.net/mmm-examples/

GenDetect is based on the StyleRank algorithm, and won the 2019 MIREX Patterns for Prediction task, achieving 99% accuracy in discriminating continuations.

Appendix D - Copyright Considerations for the MetaMIDI Dataset

We discuss considerations with respect to Canadian copyright law as it pertains to the distribution of the MetaMIDI dataset. Based on the recommendations of a copyright officer at Simon Fraser University, we limit the release the dataset to prospective users who will be using it for research.

Appendix E - StyleRank Appendix

We expand on the experiments validating StyleRank that are presented in Chapter 5. We also provide a more detailed explanation of some of the more complex features that the StyleRank uses.

1.2.3 Publications and Authorship

For each of the publications listed below, Philippe Pasquier was involved in the ideation of research questions, iterative reviews of research progress, and revising the paper content for publication.

[21] Building the MetaMIDI Dataset: Linking Symbolic and Audio Musical Data

Ens, J. & Pasquier, P. (2021). *Building the MetaMIDI Dataset: Linking Symbolic and Audio Musical Data.* In the International Society for Music Information Retrieval (ISMIR). (pp. 182-188).

Note about authorship: Jeff Ens is the first author on this paper, who assembled the the MetaMIDI Dataset, implemented the symbolic-audio metadata matching techniques, and wrote and revised the paper content for publication.

[22] CAEMSI : A Cross-Domain Analytic Evaluation Methodology for Style Imitation

Ens, J. & Pasquier, P. (2018). *CAEMSI : A Cross-Domain Analytic Evaluation Methodology for Style Imitation.* In the International Conference on Computational Creativity (ICCC). (pp. 64-71). (Best Student Paper Award)

Note about authorship: Jeff Ens is the first author on this paper, who developed the statistic methods outlined in the paper, implemented several experiments outlined in the paper which demonstrate the effectiveness of the approach, and wrote and revised the paper content for publication.

[26] Quantifying Musical Style: Ranking Symbolic Music based on Similarity to a Style

Ens, J. & Pasquier, P. (2019). *Quantifying Musical Style: Ranking Symbolic Music based on Similarity to a Style.* In the International Society for Music Information Retrieval (ISMIR). (pp. 870-877).

Note about authorship: Jeff Ens is the first author on this paper, who developed the similarity measure / ranking mechanism known as StyleRank, implemented several experiments outlined in the paper which demonstrate the effectiveness of StyleRank, and wrote and revised the paper content for publication.

[24] Improved Listening Experiment Design for Generative Systems

Ens, J. & Pasquier, P. (2020). *Improved Listening Experiment Design for Generative Systems.* 1st Conference on AI Music Creativity (AIMC).

Note about authorship: Jeff Ens is the first author on this paper, who implemented the experimental evidence which is presented in the paper, and wrote and revised the paper content for publication.

The Multi-Track Music Machine: A Generative System Designed for Co-Creative Music Composition

Ens, J. & Pasquier, P. (2022). *The Multi-Track Music Machine: A Generative System Designed for Co-Creative Music Composition*. In preparation for submission.

Note about authorship: Jeff Ens is the first author on this paper, who designed the Multi-Track Music Machine (MMM), implemented the experiments evaluating MMM in terms of plagiarism, stylistic similarity, and attribute control, and wrote and revised the paper content for publication.

[27] The Significance of the Low Complexity Dimension in Music Similarity Judgements

Ens, J., Riecke, B.E., & Pasquier, P. (2017). *The Significance of the Low Complexity Dimension in Music Similarity Judgements.* In the International Society for Music Information Retrieval (ISMIR). (pp. 31-38).

Note about authorship: Jeff Ens is the first on this paper, who conducted the research under the supervision of Bernhard Riecke and Philippe Pasquier, designed the study, ran the study, collated the findings, conducted statistical analysis on the results, and wrote and revised the paper content for publication.

[23] Discriminating Symbolic Continuations with GenDetect

Ens, J. & Pasquier, P. (2019). *Discrimiating Symbolic Continuations with GenDetect.* MIREX Patterns for Prediction Task Results (ISMIR).

Note about authorship: Jeff Ens is the first on this paper, who designed and implemented the GenDetect algorithm, and wrote and revised the paper content for publication.

1.3 Conclusion

In conclusion, we will briefly review the structure of the thesis. The narrative of the thesis begins by exploring the literature and related work for CSPG systems (Ch. 2). Then we outline contributions related to research areas A and B including the MetaMIDI Dataset (Ch. 3), CAEMSI (Ch. 4), StyleRank (Ch. 5), and recommendations for listening study design (Ch. 6). The thesis concludes with the Multi-Track Music Machine (MMM) (Ch. 7), a CSPG system trained using the MetaMIDI Dataset, with an evaluation that incorporates our previous research (StyleRank). Content provided in the appendix is not central to the thesis but provides context. As a result, it is not crucial for the reader to read the appendices in their entirety.

Bibliography

[1] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. “The million song dataset”. In: (2011).

[2] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. “Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription”. In: *International Conference on Machine Learning* (2012).

[3] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. *Deep Learning Techniques for Music Generation*. Computational Synthesis and Creative Systems. Springer International Publishing, 2019.

[4] Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. “MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer”. In: *Proceedings of the 19th International Symposium for Music Information Retrieval*. 2018, pp. 747–754.

[5] Huimin Chen, Xiaoyuan Yi, Maosong Sun, Wenhao Li, Cheng Yang, and Zhipeng Guo. “Sentiment-Controllable Chinese Poetry Generation.” In: *IJCAI*. 2019, pp. 4925–4931.

[6] Keunwoo Choi, George Fazekas, and Mark Sandler. “Text-based LSTM networks for automatic music composition”. In: *arXiv preprint arXiv:1604.05358* (2016).

[7] Ewa Dahlig. “Judgments of humans and machine authorship in real and artificial folksongs”. In: *Computing in musicology: a directory of research* 11 (1998), pp. 211–219.

[8] Ken Déguernel, Bob LT Sturm, and Hugo Maruri-Aguilar. “Investigating the relationship between liking and belief in AI authorship in the context of Irish traditional music”. In: *CREAI 2022 Workshop on Artificial Intelligence and Creativity*. 2022.

[9] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. “Jukebox: A generative model for music”. In: *arXiv preprint arXiv:2005.00341* (2020).

[10] Chris Donahue, Huanru Henry Mao, Yiting Ethan Li, Garrison W Cottrell, and Julian McAuley. “LakhNES: Improving multi-instrumental music generation with cross-domain pre-training”. In: *Proc. of the 20th International Society for Music Information Retrieval Conference*. 2019, pp. 685–692.

[11] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018, pp. 34–41.

[12] Tuomas Eerola, Tommi Himberg, Petri Toiviainen, and Jukka Louhivuori. “Perceived complexity of western and African folk melodies by western and African listeners”. In: *Psychology of Music* 34.3 (2006), pp. 337–371.

[13] Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. “Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms”. In: *arXiv preprint arXiv:1706.07068* (2017).

[14] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. “Neural audio synthesis of musical notes with wavenet autoencoders”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1068–1077.

[15] Ronald S Friedman and Christa L Taylor. “Exploring emotional responses to computationally-created music.” In: *Psychology of Aesthetics, Creativity, and the Arts* 8.1 (2014), p. 87.

[16] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. “Deepbach: a steerable model for bach chorales generation”. In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 1362–1371.

[17] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse H. Engel, and Douglas Eck. “Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset”. In: *7th International Conference on Learning Representations*. 2019.

[18] Hermann Hild, Johannes Feulner, and Wolfram Menzel. “HARMONET: A neural net for harmonizing chorales in the style of JS Bach”. In: *Advances in neural information processing systems*. 1992, pp. 267–274.

[19] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron C. Courville, and Douglas Eck. “Counterpoint by Convolution”. In: *Proceedings of the 18th International Society for Music Information*. 2017, pp. 211–218.

[20] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. “Music Transformer: Generating Music with Long-Term Structure”. In: *7th International Conference on Learning Representations*. 2019.

[21] **Jeff Ens** and Philippe Pasquier. “Building the MetaMIDI Dataset: Linking Symbolic and Audio Musical Data”. In: *Proc. of the International Symposium on Music Information Retrieval*. 2021, pp. 182–188.

[22] **Jeff Ens** and Philippe Pasquier. "CAEMSI: A Cross-Domain Analytic Evaluation Methodology for Style Imitation." In: *Proceedings of the International Conference on Computational Creativity*. 2018, pp. 64–71.

[23] **Jeff Ens** and Philippe Pasquier. "Discriminating Symbolic Continuations with GenDetect". In: *MIREX Patterns for Prediction Task Results (ISMIR)*. 2019.

[24] **Jeff Ens** and Philippe Pasquier. "Improved Listening Experiment Design for Generative Systems". In: *Proc. of the Joint Conference on AI Music Creativity*. 2020.

[25] **Jeff Ens** and Philippe Pasquier. "MMM : Exploring Conditional Multi-Track Music Generation with the Transformer". In: *arXiv:2008.06048*. 2020.

[26] **Jeff Ens** and Philippe Pasquier. "Quantifying Musical Style: Ranking Symbolic Music based on Similarity to a Style". In: *Proc. of the International Symposium on Music Information Retrieval*. 2019, pp. 870–877.

[27] **Jeff Ens**, Bernhard Riecke, and Philippe Pasquier. "The Significance of the Low Complexity Dimension in Music Similarity Judgements". In: *Proceed. of the International Symposium on Music Information Retrieval*. Vol. 18. 2017, pp. 31–38.

[28] Daniel D Johnson. "Generating polyphonic music using tied parallel networks". In: *International conference on evolutionary and biologically inspired music and art*. Springer. 2017, pp. 128–143.

[29] Kevin Joslyn, Naifan Zhuang, and Kien A Hua. "Deep Segment Hash Learning for Music Generation". In: *arXiv preprint arXiv:1805.12176* (2018).

[30] James C Kaufman, John Baer, and Jason C Cole. "Expertise, domains, and the consensual assessment technique". In: *The Journal of creative behavior* 43.4 (2009), pp. 223–233.

[31] Carolyn Lamb, Daniel G Brown, and Charles LA Clarke. "Human Competence in Creativity Evaluation." In: *ICCC*. 2015, pp. 102–109.

[32] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. "Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints". In: *Journal of Creative Music Systems* (2018).

[33] Feynman Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. "Automatic Stylistic Composition of Bach Chorales with Deep LSTM." In: *Proceedings of the International Symposium on Music Information Retrieval*. 2017, pp. 449–456.

[34] Huanru Henry Mao, Taylor Shin, and Garrison W. Cottrell. "DeepJ: Style-Specific Music Generation". In: *12th IEEE International Conference on Semantic Computing*. 2018, pp. 377–382.

[35] Matt McVicar, Raúl Santos-Rodrıguez, Yizhao Ni, and Tijl De Bie. "Automatic chord estimation from audio: A review of the state of the art". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.2 (2014), pp. 556–575.

[36] Nicholas Meade, Nicholas Barreyre, Scott C. Lowe, and Sageev Oore. “Exploring Conditioning for Generative Music Systems with Human-Interpretable Controls”. In: *Proceedings of the International Conference for Computational Creativity* (2019), pp. 148–155.

[37] David C Moffat and Martin Kelly. “An investigation into people’s bias against computational creativity in music composition”. In: *Assessment* 13.11 (2006), pp. 1–8.

[38] Olof Mogren. “C-RNN-GAN: Continuous recurrent neural networks with adversarial training”. In: *arXiv preprint arXiv:1611.09904* (2016).

[39] David Norton, Derrall Heath, and Dan Ventura. “Accounting for Bias in the Evaluation of Creative Computational Systems: An Assessment of DARCI.” In: *ICCC*. 2015, pp. 31–38.

[40] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. “This time with feeling: learning expressive musical performance”. In: *Neural Computing and Applications* (2018), pp. 1–13.

[41] Philippe Pasquier, Arne Eigenfeldt, Oliver Bown, and Shlomo Dubnov. “An introduction to musical metacreation”. In: *Computers in Entertainment (CIE)* 14.2 (2016), pp. 2–16.

[42] Ashis Pati, Alexander Lerch, and Gaëtan Hadjeres. “Learning to Traverse Latent Spaces for Musical Score Inpainting”. In: *Proc. of the 20th International Society for Music Information Retrieval Conference*. 2019, pp. 343–351.

[43] Christine Payne. “MuseNet”. In: *OpenAI* (Apr. 2019). openai.com/blog/musenet.

[44] Marcus T. Pearce and Geraint A. Wiggins. “Evaluating cognitive models of musical composition”. In: *Proceedings of the 4th international joint workshop on computational creativity*. Goldsmiths, University of London. 2007, pp. 73–80.

[45] Colin Raffel. “Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching”. PhD thesis. Columbia University, 2016.

[46] Adam Roberts, Jesse H. Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. “A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music”. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 4361–4370.

[47] Bob L Sturm. “The state of the art ten years after a state of the art: Future research in music information retrieval”. In: *Journal of new music research* 43.2 (2014), pp. 147–172.

[48] Bob L Sturm and Oded Ben-Tal. “Taking the models back to music practice: Evaluating generative transcription models built using deep learning”. In: *Journal of Creative Music Systems* 2.1 (2017).

[49] Ruben Rodriguez Torrado, Ahmed Khalifa, Michael Cerny Green, Niels Justesen, Sebastian Risi, and Julian Togelius. “Bootstrapping conditional gans for video game level generation”. In: *2020 IEEE Conference on Games (CoG)*. IEEE. 2020, pp. 41–48.

[50] Christian Walder. “Modelling symbolic music: Beyond the piano roll”. In: *Asian Conference on Machine Learning*. 2016, pp. 174–189.

[51] Jacob Westfall, David A Kenny, and Charles M Judd. “Statistical power and optimal design in experiments in which samples of participants respond to samples of stimuli.” In: *Journal of Experimental Psychology: General* 143.5 (2014).

[52] Li-Chia Yang and Alexander Lerch. “On the evaluation of generative models in music”. In: *Neural Computing and Applications* (2018), pp. 1–12.

Chapter 2

Related Work on Symbolic Polyphonic Music Generation

2.1 Introduction

Many researchers have sought to develop creative systems capable of generating musical material. Research involving musical meta-creative (MuMe) [75] systems is a subfield of computational creativity, an emerging field of research which focuses on the simulation or replication of creativity using a computer [20]. In contrast to many problems in AI, where the notion of an optimal solution is well-defined, when generating a creative artifact, such as music, there is no clear best solution [75]. Although this poses difficulties when developing and evaluating these systems, recent advancements in Deep Learning have resulted in state-of-the-art MuMe systems [60] that are human competitive in certain contexts.

In what follows, we will summarize key developments involving corpus-based symbolic polyphonic music generation (CSPG) systems. Before moving any further, it is necessary to clarify what is meant by the terms corpus-based, symbolic, and polyphonic. In contrast with a rule-based system, where relevant domain knowledge is encapsulated in a set of explicit rules [35], a corpus-based system extracts knowledge from data. Here, we consider any non-empty set of data to be a valid corpus, so models that are trained on a single musical excerpt [41] are included. The term symbolic music refers to representations where musical events, such as notes, are explicitly encoded [23]. Note that this differs from raw audio, which is a continuous stream of scalar values, and contains no explicit information about what notes are sounding.

Although many of the contributions outlined throughout this thesis are broadly related to the topic of symbolic music generation (SMG), we restrict our summary of related work to CSPG systems, ignoring non-corpus-based and non-polyphonic systems for the following reason. The Multitrack Music Machine (MMM) (Ch. 7) is a CSPG, and including systems that are tangentially related to the design of MMM (i.e. non-corpus-based, non-polyphonic) would do little but provide distraction. Furthermore, since there is a large body of research discussing CSPG systems, let alone SMG systems, there is a substantial amount of ground which must be covered whilst imposing this constraint anyways.

This chapter is organized as follows. In Section 2.2, we outline the typology that we use to organize CSPG systems. Then the specifics of each system will be outlined. To conclude, we discuss the current challenges and propose avenues for future research (Section 2.6).

2.2 Typology

To organize our discussion, we categorize CSPG systems using several factors: the computation model (2.2.1); the musical texture of the output (2.2.2); and the data representation employed (2.2.3). When available, we also provide information on the dataset (2.2.4), analytic evaluation methods (2.2.5), listening test (2.2.5), and a link to musical examples generated by the system. Table 2.1 lists the CSPG systems discussed throughout the chapter, and additional details on the typology are detailed in the rest of this section.

	System	**Ex.**	**Model (2.2.1)**	**Texture (2.2.2)**	**Rep. (2.2.3)**	**Dataset (2.2.4)**	**Metric (2.2.5)**	**Listening Test (2.2.5)**
A1	Padilla et al.[71]	-	Markov chain	Fixed	Univar. Sequence	Palestrina ⋆	Information Content	55 participants †
A2	Whorley et al.[97]	-	Markov chain	Fixed	Univar. Sequence	English Hymns (100) ⋆	12 custom metrics	-
A3	Eigenfeldt et al.[28]	-	Markov chain	Hom.	Univar. Sequence	Assorted Genres ⋆	-	-
A4	FlowComposer[73]	link	Markov chain	Hom.	Univar. Sequence	Lead Sheet Database	-	-
A5	Ponsford et al.[79]	-	Markov chain	Hom.	Univar. Sequence	17th Century (84) ⋆	-	-
A6	Racchmaninof[17]	link	Markov chain	Poly.	Univar. Sequence	Bach Chorales (DT2)‘ Chopin ⋆	-	16 non-musician 15 musician
B1	Allan et al.[2]	link	HMM	Fixed	Graph	Bach Chorales (DT2)	-	-
B2	i-Ring[59]	-	HMM	Hom.	Graph	Unspecified (150) ⋆	-	10 participants
B3	Kaliakatsos et al.[53]	-	HMM	Hom.	Graph	Subset of Bach Chorales (DT2)	Pitch Class Similarity Root Similarity Exact Matches	-
B4	Paiement et al.[72]	-	HMM	Hom.	Graph	Lead Sheets (47) ⋆	Log-loss	-
B5	Raczynski et al.[80]	-	HMM	Hom.	Graph	Wikifonia (DT1)	Log-loss	-
B6	Simon et al.[87]	link	HMM	Hom.	Graph	Lead Sheets (298) ⋆	-	30 participants
C1	CoCoNet[46]	link	CNN	Fixed	Multivar. Vector	Bach Chorales (DT2)	Log-loss	96 comparisons †
C2	C-RBM[57]	link	CNN	Poly.	Multivar. Vector	Mozart/Batik (DT9)	Information Rate	-
D1	Hild[42]	-	FFNN	Fixed	Multivar. Sequence	Bach Chorales (DT2)	-	-
E1	MuseGan[27]	link	GAN	Poly.	Multivar. Vector	Lakh Midi Dataset (Clean) (DT8)	Qualified Note Rate Polyphonicity Tonal Distance Log-loss	20 participants

	System	Ex.	Model (2.2.1)	Texture (2.2.2)	Rep. (2.2.3)	Dataset (2.2.4)	Metric (2.2.5)	Listening Test (2.2.5)
E2	C-RNN-GAN[66]	link	GAN	Poly.	Multivar. Sequence	Classical (3697) ⋆	Scale consis-tency Note range Tone count Velocity range Polyphony 3-tone repeti-tion	-
F1	DeepBach[38]	link	RNN	Fixed	Multivar. Sequence	Bach Chorales (DT2)	Log-loss	1272 participants †
F2	Josyln et al.[52]	-	RNN	Hom.	Multivar. Sequence	Nottingham (DT4)	Log-loss	-
F3	DeepJ[62]	-	RNN	Poly.	Multivar. Sequence	Classical ⋆	Log-loss	20 participants
F4	Johnson[50]	link	RNN	Poly.	Multivar. Sequence	Bach Chorales (DT2) Nottingham (DT4) Musedata (DT5) Piano-Midi.de (DT6)	Log-loss	-
F5	LSTM-DBN[95]	-	RNN	Poly.	Multivar. Sequence	Bach Chorales (DT2) Nottingham (DT4) Musedata (DT5) Piano-Midi.de (DT6)	Log-loss	-
F6	RNN-DBN[32]	-	RNN	Poly.	Multivar. Sequence	Bach Chorales (DT2) Nottingham (DT4) Musedata (DT5) Piano-Midi.de (DT6)	Log-loss	-
F7	RNN-RBM[8]	link	RNN	Poly.	Multivar. Sequence	Bach Chorales (DT2) Nottingham (DT4) Musedata (DT5) Piano-Midi.de (DT6)	Log-loss	-
F8	BachBot[60]	link	RNN	Fixed	Univar. Sequence	Bach Chorales (DT2)	Log-loss	2336 participants †
F9	Choi et al.[15]	link	RNN	Hom.	Univar. Sequence	Jazz Lead Sheets ⋆	Log-loss	-
F10	Meade et al.[64]	link	RNN	Poly.	Univar. Sequence	Piano-e-Competition (DT3)	-	-
F11	PerformanceRNN[69]	link	RNN	Poly.	Univar. Sequence	Piano-e-Competition (DT3)	Log-loss	-

	System	Ex.	Model (2.2.1)	Texture (2.2.2)	Rep. (2.2.3)	Dataset (2.2.4)	Metric (2.2.5)	Listening Test (2.2.5)
F12	Walder et al.[96]	link	RNN	Poly.	Univar. Sequence	Bach Chorales (DT2) Nottingham (DT4) Musedata (DT5) Piano-Midi.de (DT6)	Log-loss	-
G1	MuseNet[76]	link	Transformer	Poly.	Univar. Sequence	Assorted Genres ⋆	-	-
G2	Music Transformer[47]	link	Transformer	Poly.	Univar. Sequence	Maestro Dataset (DT10)	Log-loss	180 comparisons †
G3	**Multi-Track Music Machine** (Ch.7)	link	Transformer	Poly.	Univar. Sequence	MetaMIDI Dataset	See Section 7.6	-
G4	MusIAC[36]	-	Transformer	Poly.	Univar. Sequence	Lakh MIDI (DT7)	-	-
G5	LakhNES[25]	link	Transformer	Poly.	Univar. Sequence	Lakh MIDI (DT7) NES Music Database	Log-loss	180 participants
H1	MidiVAE[9]	link	VAE	Poly.	Multivar. Sequence	Jazz / Pop / Classical ⋆	Style Classifier Distance	-
H2	MusicVAE[84]	link	VAE	Fixed	Univar. Sequence	Assorted Genres (1.5 million) ⋆	Log-loss	192 comparisons †
I1	MorpheuS[41]	link	VNS	Poly.		-	Time Complexity	-

Table 2.1: An overview of CSPG systems where Univar., Multivar., and VNS denote univariate, multivariate, and variable neighbourhood search, respectively. The example (Ex.) column provides links to audio samples. When available, the size of a dataset is shown in parenthesis. In cases where authors created their own dataset, demarcated by a ⋆, a general description is provided instead of the dataset name. The metric column lists the quantitative measures used to evaluate the system. The listening test column indicates the number of participants or comparisons in the study, where † indicates that the artifacts generated by the system are human competitive. For each column, more details are available in the corresponding section specified in parentheses.

2.2.1 Model

Any CSPG system must employ some sort of model that is capable of extracting the relevant information from a corpus of musical data after undergoing some sort of training process. Although neural networks are the typical choice, other models have been employed, including Markov chain, Hidden Markov Models, and Variable Neighbourhood Search. In many cases, selecting a particular model, imposes constraints on the representations that can be used to represent musical data. For example, using a Markov chain requires data to be represented as a univariate sequence of discrete tokens. Early work involving neural networks used Feed-Forward Networks (FNN) and Recurrent Neural Networks (RNN) to model musical material. However, in recent years, CSPG systems have been developed that use a variety of neural network architectures, including Convolutional Neural Networks (CNN), Generative Adversarial Networks (GAN), Variational Auto-Encoders (VAE) and attention-based neural networks such as the Transformer.

2.2.2 Musical Texture

In general, there are three types of musical texture: Monophonic, which consists of a single melodic line with no accompaniment; Homophonic, which consists of a single melody and chordal accompaniment; and Polyphonic, which contains multiple melodic voices which exhibit some degree of independence from each other [6]. To be clear, in a polyphonic texture, melodic voices are not completely independent from each other, as each voice is dependant on the pitch and rhythmic content of other voices. However, in contrast to homophonic textures, where the chordal accompaniment is comprised of multiple voices that share the same rhythm, the voices in a polyphonic texture will employ different rhythms. Notably, it is common for the term Polyphonic to refer to all musical textures that are not monophonic [24]. To account for this ambiguity, and the fact that some systems could be construed as either generating homophony or polyphony (D1, A3, B1, A2), we adopt the broad definition of polyphony (anything not monophonic) as a condition for inclusion in this survey, yet retain the specific definitions for the convenience of categorization.

2.2.3 Representation

We identify four different approaches to representing musical material: univariate sequences, multivariate sequences, multivariate vectors and graphs. Note that we consider the manner in which musical material is represented from the perspective of the model (2.2.1).

Univariate Sequence

Using a univariate sequence involves providing a series of scalar values to the model $(x_0, x_1, ...x_n)$ one by one, and conditioning the generation of each scalar value (x_t) on the series of scalar values that have previously been provided $(x_0, x_1, ..., x_{t-1})$. Typically, representing music as a univariate sequence involves constructing a sequence of tokens (i.e. integers) using a discrete alphabet consisting of n distinct tokens which each correspond to a specific musical event. For example, 128 distinct NOTE_ON tokens may be used to indicate the onset of each possible MIDI pitch, and a TIME_SHIFT token may be used to indicate some passage of time. Some systems (F11, G2) use a single TIME_SHIFT token (F8), while others use n distinct TIME_SHIFT tokens to represent the passage of different lengths of time.

Multivariate Sequence

The only difference between multivariate sequences and univariate sequences, is that each item in a multivariate sequence is comprised of multiple scalar values (multivariate) rather than a single scalar value (univariate). For example, Mogren et al. (E2) represent each note using four real valued scalars: note length, frequency, velocity and time-delta (in seconds), and train an RNN using a multivariate sequence of continuous 4-dimensional vectors.

Multivariate Vector

Using a multivariate vector involves providing multiple scalar values to the model, generating a prediction using these values exclusively. One example of a system which uses this representation is CoCoNet (C1), which takes an incomplete piano roll as input and generates the missing parts of the piano roll (filling in the blanks).

Graph

Other systems (B1, B6, B3, B4, B5) adopt a graph representation, where musical events are the nodes, and edges denote relationships between events. For example, Allan and Williams (B1) represent fixed voice polyphony using a hidden Markov model (HMM), where each melody note is an observed node, all notes in other voices are hidden nodes, and each edge represents the conditional probability of a transition between states.

Ambiguity of the Piano Roll

Since we categorize systems based on the manner in which musical material is processed by the model, we are able to make distinctions between systems that use a piano roll representation based on the way the information is actually processed. A piano roll is a boolean matrix $x \in \{0, 1\}^{T \times P}$, where T is the number of time-steps and P is the number of pitches. Typically $P = 128$, allowing the piano roll to represent all possible MIDI pitches. Each column represents the pitches that are sounding at a particular time-step, which is an even subdivision of the beat, as discussed in 2.2.4. Some models process several time-steps of piano roll simultaneously (multivariate vector) (C1, C2, E1), while others process each time-step conditioned on previous time-steps (multivariate sequence) (F7, F6, F5, F4, F3). Furthermore, by traversing the piano roll in column-major or row-major order, it can be represented as a univariate sequence of boolean values, and provided to a model that is designed to learn univariate sequences (RNN, Transformer). However, this is clearly a sub-optimal approach due to the extremely long sequences required to represent musical material.

2.2.4 Data

Datasets

Table 2.2 provides an overview of the datasets used by the systems discussed in this chapter. Although a dataset may have been cleaned and pre-processed, this does not guarantee that it is free of problems. For example, large datasets like the Lakh MIDI Dataset (DT7) may contain corrupt MIDI files that are unusable. Another example, discovered by Sturm [88], is the Jukedeck MIDI versions [1] of the Nottingham Dataset (DT4), which do not correctly encode the repetition structure of the tunes as originally notated in the ABC format. Ultimately, the quality of a generative system is limited by the quality of the data, so one should take care to audit the quality of datasets being used.

Storage Specifications and Standards

A common format for representing symbolic music is the Musical Instrument Digital Interface (MIDI) protocol [67], which was developed in the early 1980's to standardize the representation of musical performance information in a digital format [40]. MIDI files are simply a series of messages, and notes are represented using two types of MIDI message: the `NOTE_ON`, and `NOTE_OFF` messages, which both consist of a channel number, a pitch and a velocity (loudness). There are a variety of messages that can control various musical features, such as the tempo, time-signature markings, and even changes in instrumentation. Timing information is specified in ticks, with a single tick defined as a fraction of a quarter note beat. The number of ticks per quarter note beat is specific to each MIDI file, and is specified in the MIDI file header. In the case that data is not available in MIDI format, code libraries are readily available that produce MIDI files from a variety of formats including ABC, musicXML and Humdrum [22].

Notably, the MIDI format is not without its limitations, and may not be the ideal for all use-cases, especially in cases where a encoding format has been specifically developed for a particular type of music. For example, ABC notation was designed to encode folk and traditional tunes, providing an efficient method to notate structural repetition, which is not possible using MIDI. Another example is a lead sheet, which is typically used to represent jazz music, and is comprised of a melody and chord symbols denoting the harmonic accompaniment.

Pre-processing

When working with symbolic musical data, one common form of data pre-processing involves quantizing rhythmic information to an even subdivision of the beat. Often, note onsets and offsets are quantized to 4 subdivisions per beat (F7, F6, F5, F8, F1), resulting in a sixteenth note resolution, however, some models (F11, E1, G2) have been trained on data quantized at a much higher resolution. When an appropriate quantization threshold is selected, such as a sixteenth note resolution for

[1]https://github.com/jukedeck/nottingham-dataset

Bach Chorales, the musical composition will not be affected. However, the small rhythmic deviations from the metric grid that are found in an expressive musical performance will be lost unless rhythmic information is quantized at an extremely high resolution.

Many CSPG systems generate compositional material which can optionally be performed by an expressive performance system [11]. However, some CSPG systems are designed to generate both the compositional material and the expressive performance (F11, G1, G2, F3), which necessitates quantization of rhythmic information at a much higher threshold, retaining the expressive micro-timing of the performer. In addition, these systems integrate quantized velocity information, capturing variations in loudness which are characteristic of compelling musical performances. Another type of pre-processing involves transposing pitches so that the key of each piece is in either A minor or C major (F7, F2).

Data Augmentation

Data augmentation applies various manipulations to the data to increase the diversity and amount of data. Even simple data augmentation strategies for image based corpora that involve cropping, padding, vertical flipping, horizontal flipping, and rotation can be very effective [58, 55]. For musical data, one common data augmentation technique is transposition, which usually involves randomly transposing each piece ± 6 semitones (F12, G5, E1), as this enables the trained model to generate music in any key. In the process of training the Performance RNN, Oore et al. (F11) augmented the duration of all notes within a piece by stretching/compressing notes up to 10%.

	Dataset	Description	Expressive Parameters	Size	Format	Link
DT1	Wikifonia ⋆	Assorted lead sheets.	-	2,000	-	-
DT2	Bach Chorales	Bach chorales corpus available in music21.	-	382	musicXML	https://web.mit.edu/music21/
DT3	Piano-e-Competition	Performances of classical piano compositions recorded on a Disklavier at the Piano-e-Competition.	velocity, timing	1,400	MIDI	http://www.piano-e-competition.com
DT4	Nottingham	Folk tunes.	-	1,000	MIDI	https://github.com/jukedeck/nottingham-dataset
DT5	MuseData	Assorted classical compositions.	-	881	MIDI	http://musedata.org/
DT6	Piano-Midi.de	Assorted classical compositions from 25 composers.	-	130	MIDI	http://www.piano-midi.de/
DT7	Lakh Midi Dataset[81]	De-duplicated MIDI files, including a wide variety of genres.	-	174,154	MIDI	https://colinraffel.com/projects/lmd/
DT8	Lakh Midi Dataset (Clean)[81]	A subset of the Lakh Midi Dataset with filenames which indicate their artist and title.	-	21,425	MIDI	https://colinraffel.com/projects/lmd/
DT9	Mozart/Batik[98]	Note-wise matched dataset containing Roland Batik's performances and the corresponding scored material.	velocity, timing	37	In-house format	-
DT10	Maestro Dataset[39]	Paired audio and MIDI recordings from ten years of International Piano-e-Competition.	velocity, timing	1184	MIDI + Audio	https://magenta.tensorflow.org/datasets/maestro
DT11	Palestrina[22]	Palestrina masses corpus available in music21.	-		musicXML	https://web.mit.edu/music21/
DT12	Lead Sheet Database ⋆[70]	Assorted lead sheets.	-	10,000	-	-
DT13	NES Music Database[26]	NES video game music	-	5,278	-	https://github.com/chrisdonahue/nesmdb

Table 2.2: Datasets that are used by CSPG systems. ⋆ denotes datasets that are no longer available. The expressive parameters column indicates if the dataset contains attributes related to the expressive performance of a musical excerpt such as velocity, and note articulation (timing).

2.2.5 Evaluation of CSPG Systems

Since CSPG systems can more broadly be considered creative systems, we can apply evaluation methodologies designed for creative systems to CSPG systems. To frame our discussion we consider four perspectives from which a creative system can be evaluated. These perspectives were initially identified by Rhodes [82] and have been contextualized for the evaluation of creative computational systems by Lamb et al. [56]. The four perspectives include: the *Person*, which focuses on the creative capacity of a system; the *Process*, which concerns the specific actions that a system takes to produce a creative artifact; the *Product*, which analyzes artifacts produced by a system; and *Press*, which considers the cultural perception of a system.

Since evaluating a system from the *Person* perspective involves assessing its creative capacity, Lamb et al. suggest that methodologies designed to measure human creativity are most appropriate. Although computational systems can pass these types of tests, this can largely be attributed to limitations with the evaluation methodology, and the fact that these systems are designed specifically to do nothing more than pass a particular test. For example, comRAT-C [68] can solve a Remote Associates Test by using a database of common bi-grams, but it is not capable of performing any creative tasks beyond this. Furthermore, since CSPG systems are predominantly domain-specific, and do not exhibit domain-general creativity, evaluation from the *Person* perspective is rarely employed.

The *Process* perspective integrates theories about how creative products are made. Clearly, this requires a cogent understanding of how a system actually produces a generative artifact. Given the inherent complexity of most neural network based systems, it would be extremely difficult to meaningfully describe the processes by which these systems produce generative artifacts. Understandably, evaluation via the process perspective is much more amenable to rule-based systems, where processes are explicitly laid out in a set of rules, rather than so-called "black-box" systems.

It is very common to evaluate a system from the *Product* perspective, allowing an evaluation to be formed based on attributes of the generated artifacts. There is a general consensus that both the novelty, and quality of the generated artifacts should be taken into consideration [83]. A desirable creative system will generate artifacts which are both novel and exhibit quality, as novelty without quality is often perceived as randomness, and quality without novelty often results in plagiarism of pre-existing material. Typically, a quantitative methodology is applied either using humans or computational techniques, however, informal qualitative feedback has been used to evaluate the output of some CSPG systems (F11).

The *Press* perspective, which evaluates a system based on its social effect, is inherently linked to the other perspectives, as evaluation from the *Person*, *Process* or *Product* perspective is unavoidably biased by our own cultural context. Of course, evaluation via the *Press* perspective can also be conducted by observing engagement with the system across different media platforms, and materials published via traditional media. However, it is worth noting that large research institutions can dominate coverage, which should be taken into account when evaluating from the *Press* perspective. Furthermore, since media coverage and social media engagement can build slowly over longer periods of time, it can be difficult or infeasible to conduct an evaluation from this perspective in many circumstances. Of course, it is also possible to evaluate a system from multiple perspectives simultaneously. One notable example is SPECS[51], which requires participants to rate a system based on 14 components of creativity, incorporating both the *Process* and *Product* perspectives. However, in what follows, we discuss the evaluation of CSPG systems from the *Process* and *Product* perspective, as these perspective are more frequently employed by researchers. Notably, the contributions outlined in the later chapters adopt the *Product* perspective exclusively.

Process Perspective Evaluation Methods

We begin by describing evaluation methods from the Process perspective. Note that these methods are quite generic and are designed to accommodate any creative system. Colton et al. propose FACE [19], an evaluation framework that is based on the concept of generative acts. They describe a variety of generative acts, including creating a concept, an expression of a concept, an aesthetic measure, framing information, as well as creating methods to generate any of the former generative acts. The authors propose that creative systems can be evaluated by adding up the total number of generative acts performed by a system, or by ranking the importance of different generative acts in order to compare the entire creative process. Admittedly, this framework could use further specification to enable consistent application in various evaluation scenarios, however, it still provides a useful conceptual framework for evaluation from the process perspective.

Ventura et al. [93] proposes a seven level hierarchy of creative systems, which includes the following levels: Randomization, Plagiarization, Memorization, Generalization, Filtration, Inception, Creation. At the bottom of the hierarchy, are systems that are only capable of random generation. In contrast to random generation systems, which do not learn from any data, the remaining six levels in the hierarchy integrate knowledge from a dataset in some manner. Memorization systems are distinct from plagiarism systems, as errors in the memorization process introduce novelty. Generalization systems are capable of generating new material, while filtration systems are capable of filtering out undesirable generations. The final two levels, inception and creation, require the system to use additional knowledge that is not contained in the dataset. Creation systems, the highest level in the hierarchy, should be capable of perceiving their context and letting that influence generation. One of the disadvantages of this approach is that the levels are quite coarse. For example, many of the systems we describe in this chapter would be considered generalization systems, leaving us without a way to make meaningful comparisons between systems.

Process and Product Perspective Evaluation Methods

As was noted earlier, the SPECS methodology [51] incorporates both the *Process* and *Product* perspectives. SPECS is based on 14 components of creativity, which were derived from keywords aggregated from a large collection of papers discussing creativity. The methodology involves 3 steps. First, researchers must define creativity, which involves assigning varying levels of importance to the 14 components depending on the specifics of the creative domain or task. The second step involves determining the exact standards by which the system(s) will be evaluated, informed by the decisions made in step 1. The final step involves collecting feedback and ratings from participants in accordance with the standards defined in step 2. Although this approach is more time consuming than others, as the standards must also be determined by the researcher, it is capable of providing a more detailed picture of the strengths and weaknesses of the systems being evaluated.

Product Perspective Evaluation Methods with Human Participants

Evaluation of a CSPG system from the product perspective often involves collecting feedback from human participants who listen to excerpts generated by the system. The Consensual Assessment Technique (CAT) [3] averages domain experts independent assessment of the creativity of a product. Although the CAT has been used to evaluate generative music systems [77, 17], the difficulty of collecting assessments from domain experts has limited widespread use. A modified Turing test [90] has been used frequently in practice, where participants are asked to discriminate between computer-generated and human-composed musical excerpts. Although this experimental methodology is commonly referred to as a Turing test throughout related literature, this is technically incorrect [4]. One major weakness of this approach, is that participants may focus on identifying errors, rather than focusing on the general stylistic characteristics of the musical excerpts [4]. In order to compare multiple systems, participants can rank randomly paired musical excerpts from two different sources, where one or more of the excerpts is generated by a system [47]. Once the rankings have been collected, the number of wins can be counted for each distinct source (human-composed, system A, system B, etc.).

It is also possible to create evaluation methodologies based on various physiological measurements. One approach [13] uses electroencephalography (EEG), and involves training a linear model to predict ratings based on EEG data. The ground truth data used to train the linear model is collected while participants listen to musical excerpts. In order to produce musical excerpts of varying quality, human composed musical excerpts are randomly distorted by replacing a percentage of notes with random notes. Agres et al. discuss several physiological measurements that have been used in the context of music research and can potentially be employed to evaluate generative music systems [1]. These include measurements of pulse, Galvanic skin response, eye-tracking and motion capture.

Product Perspective Analytic Evaluation Metrics

The computational evaluation of systems from the *Product* perspective involves comparing measurable attributes of the training data C against a large collection of generated excerpts G. In contrast to human-based evaluation methodologies, where the number of musical excerpts that can be feasibly evaluated is quite small (often less than 20), computational methods can handle a much larger number of excerpts, reducing the chance of errors in evaluation due to sample size. However, human perception is often considered the gold-standard for evaluation of CSPG systems, and since there is no guarantee that analytic evaluation methods correlate with human perception, they must be validated before widespread use.

The simplest approach involves a set of metrics $m = \{m_1, ..., m_k\}$, which given a musical excerpt x produce a single scalar value $m_i(x)$. Then the mean and standard deviation of the distributions $[m_i(c) : c \in C]$ and $[m_i(g)] : g \in G]$ can be compared for each m_i in m. This approach has been used to evaluate several systems (E1, E2, A2), however, a standard set of metrics has not yet been established. CAEMSI, a domain-agnostic evaluation methodology for style-imitation, provides statistical tests to determine if the distributions $[D(c_i, c_j) : c_i \in C, c_j \in C, i \neq j]$, $[D(g_i, g_j) : g_i \in G, g_j \in G, i \neq j]$, and $[D(c, g) : c \in C, g \in G]$ are equivalent or different, where D is an arbitrary distance metric [48]. Yang et al. take the distance between $[||f_k(c_i) - f_k(c_j)||^2 : c_i \in C, c_j \in C, i \neq j]$ and $[||f_k(c) - f_k(g)||^2 : c \in C, g \in G]$ for each feature f_k, as a measure of stylistic similarity [99]. They propose 9 features (f_k), which include a pitch class histogram, note length histogram, and a note length transition matrix. StyleRank [49] uses a large collection of features and a random forest embedding technique to rank each musical excerpts based on their similarity to C. Notably, experimental evidence demonstrates that StyleRank correlates with human perception of musical similarity.

2.3 Markov Models

A Markov model [31] is a stochastic model based on the Markov assumption, which states that given the present state, future states are independent from past states. Here, we discuss the use of two Markov models, Markov chains and hidden Markov models, to generate polyphonic content.

2.3.1 Markov Chain

A Markov chain [31] is a simple Markov model where the state of the model is a random variable that changes through time. Concretely, given a sequence $x = \{x_0, x_1, ..., x_n\}$, a d^{th}-order Markov assumption implies that $p(x_t|x_0, x_1, ..., x_{t-1}) = p(x_t|x_{t-d}, ..., x_{t-1})$. As a result, given a second-order Markov model trained on a sequence $x = babcabd$, $p(c|ab) = .5$ is simply the occurrence count of the tri-gram abc divided by the occurrence count of tri-grams starting with ab. A variable-order Markov model (VOMM) is an extension of the Markov chain, which varies the order of the assumption based on the conditional probability components found in the data.

A multiple viewpoint system (MVS) [21] is a collection of Markov chains $m = \{m_1, ..., m_n\}$, where each m_i models a distinct viewpoint. Viewpoints can be vertical or horizontal. For example, a sequence of note durations could be considered a single horizontal viewpoint. Vertical viewpoints can be used to enforce harmonic constraints between simultaneously sounding notes. To generate music, the predictions from each Markov chain m_i are combined, often using a weighted arithmetic or geometric approach [78].

Fixed Voice Polyphony with Univariate Sequences

Padilla and Conklin (A1) generate two voice counterpoint with an MVS. A variety of horizontal and vertical viewpoints are incorporated into the model, including pitch, contour, relative pitch position within the dominant scale, and vertical interval. Patterns extracted from a template piece specify global repetition across multiple viewpoints, and a backtracking algorithm is used to generate musical material that respects these constraints. Whorley and Conklin (A2) use an MVS to generate four part harmony given a melody. Both vertical (inter-voice) and horizontal (intra-voice) viewpoints are used, which are selected objectively by minimizing the cross-entropy measured via cross-validation on the corpus.

Homophony with Univariate Sequences

Eigenfeldt and Pasquier (A3) use a VOMM to generate chord sequences. The users preferences are incorporated through additional factors that condition the generation such as harmonic complexity, chord transition tension, and the bass-line contour. Ponsford et al. (A5) use a Markov chain, trained on seventeenth century music, to generate chord sequences. The Flow Composer system (A4), generates lead sheets using a metrically constrained Markov chain [85], which ensures that generated sequences adhere to a particular time signature. Musical material is generated in two steps, first the chord sequence is generated, and then the melody is generated, conditioned on the chord sequence.

Polyphony with Univariate Sequences

The Racchmaninof (A6) system uses a first-order Markov chain to generate polyphonic music. Each chord segment is represented by two pieces of information: the beat position relative to the bar in which the note/chord/rest occurs, and the pitches relative to the global tonic. A point set compression algorithm, SIACT [18], is used to extract a template that delineates all occurrences of repetition within a single piece, which is used to constrain generation. Since these repetition structures can be nested, generation starts with the smallest repeated fragment. In order to fill in the space X_i between two generated segments X_{i-1} and X_{i+1}, a forward and backward sequence are generated and a join is made by randomly selecting from their intersection points. In the case that there are no intersections, X_{i-1} and X_{i+1} can be partially overwritten until a valid intersection point is found. The authors train the model on a collection of Chopin's mazurkas and a collection of Bach chorales. A listening test indicated that participants were no better than random at identifying generated Bach, however, a large majority were able to identify generated Chopin.

2.3.2 Hidden Markov Models

A hidden Markov model (HMM) [5] is a probabilistic graphical model which allows for the prediction of a set of hidden variables (X) based on a set of observed variables (Y). In order to compute the hidden states, three things must be known: the *transition* probability $p(X_j|X_i)$, which is the probability of transitioning to a new state X_j given the current state X_i is known; the *emission* probability $p(Y_j|X_i)$, which is the probability of transitioning to an observed state Y_j given the hidden state X_i is known; and the *prior* probability $p(X_i)$ of each hidden state. Given a set of observed states, the maximum probability set of hidden states can be calculated using the Viterbi algorithm [94].

Fixed Voice Polyphony with HMMs

Allan and Williams (B1) generate Bach chorales with a pair of HMMs. In the harmonization HMM, the observed states are the melody notes and the hidden states are the intervals between the melody note and each respective voice (alto, tenor bass). There is an additional hidden variable for each chord, which describes the chord function (e.g. Tonic, Dominant). To construct the ornamentation HMM, the alto, tenor, and bass voices are observed variables at each downbeat, and the following three sixteenth notes of each beat are hidden states. In order to sample from the model, the harmonization HMM is sampled first, using a backwards sampling procedure, and then the ornamentation HMM is sampled using the predictions from the harmonization HMM as visible states. As a result, the expressive capacity of the combined model is similar to BachBot (F8). They evaluate the system informally, and concluded that the maximum probability harmonizations, found using the Viterbi algorithm, are high quality. However, the backwards sampled harmonizations are of lower quality, featuring an inordinate amount of large leaps.

Homophony with HMMs

Lee et al. propose i-Ring (B2), an HMM trained with a dataset of 150 MIDI files, which infers the optimal chord sequence (hidden variables) for the provided melody (observed variables). Similarly, Simon et al. (B6) use a HMM to generate a sequence of chords accompanying a melody, given a collection of 298 lead sheets. Both authors conduct a listening test with results that affirm the quality of the generated harmonizations. Paiement et al. (B4) harmonize melodies with a hierarchical HMM, accounting for longer-term dependencies between chords, that is trained on a set of lead sheets. Another system (B5) interpolates between three trained models: a chord bi-gram HMM, which models chord transitions under a first order Markov assumption; a tonality HMM, which models the probability of a chord given a tonality label (e.g. C-major); and a melody HMM, which models the probability of a chord given a melody note. This system was trained on a collection of lead sheets sourced from Wikifonia. Another system (B3), uses an HMM to generate Bach chorale harmonizations constrained by intermediate anchor chords, providing an additional degree of control over the generation process.

2.4 Neural Networks

Neural networks [33], which were inspired by the structure of a human brain [63], are comprised of computational units (neurons) arranged into layers, with connections between different layers.

Although they were inspired by the structure of a human brain, they are not designed to be biologically realistic models. According to the Universal Approximation Theorem [45], any continuous function on a bounded set can be approximated within a given accuracy by a trained neural network. Typically, training a neural network involves optimizing a set of parameters θ with respect to a loss function $\mathcal{L}$ using gradient descent [12]. For the interested reader, Goodfellow et al. [33] provide a detailed account on neural network techniques and architectures.

2.4.1 Feed Forward Networks

In a feedforward neural network (FFNN), inputs flow directly through the network to the output [33].

Fixed Voice Polyphony with Multivariate Vectors

Hild et al. (D1) developed a system, comprised of three FFNNs, that generate Bach chorale harmonizations. Each FFNN accepts a multivariate vector as input. The first network predicts the chord C_t given the previous three chords $(C_{t-3}, C_{t-2}, C_{t-1})$ and the entire melody. The next FFNN predicts the alto and tenor voices, given the chords and the melody. Since the bass note is included in the chord representation, it was already generated by the first FFNN. The final FFNN generates eighth note ornamentations from the output of the second FFNN.

2.4.2 Convolutional Neural Networks

Convolutional neural networks (CNN) [30] are designed to process a tensor of shape (c, N_1, N_2), where c is the number of channels. CNNs are a multi-layer network comprised of two distinct types of layers: convolutional layers, and pooling layers. A convolutional layer convolves a set of l 2-dimensional filters across the last two dimensions (N_1, N_2) of the input. A pooling layer applies a down-sampling operation, partitioning the last two dimensions into a set of non-overlapping rectangular windows, and outputting the representative value for each window. Typically, the maximum value is selected from the values in each window, a process which is known as max-pooling. CNNs are shift invariant, which makes them adept at capturing features irrespective of their location within an input. As a result, CNNs are often used for image processing [55], however, as we will discuss in this section, they have also been successfully applied to the task of generating symbolic polyphonic music (C2, C1, E1).

Fixed Voice Polyphony with Multivariate Vectors

CoCoNet (C1) is a CNN, trained under the orderless NADE framework [91], that is trained using the Back Chorales dataset (DT2). For a simplified explanation of the orderless NADE framework, consider a D-dimensional vector x, and an arbitrary model m. The goal is to train m to model $p(x_{O_j}|x_{O_{<j}})$ for any j and any random ordering O of the set $\mathbb{I}_D = \{i|0 \leq i < D\}$. Note that $O_{<j}$ denotes the first j elements (zero indexed), and $O_{\geq j}$ denotes the last $D - j$ elements in the ordering O. To accomplish this, m is trained to predict $x_{O_{\geq j}}$ given $x_{O_{<j}}$ for a uniformly sampled ordering

O and index j. Here, m is a deep CNN consisting of 64 layers. Each convolution is padded so that the output is the same shape as the input, and batch normalization is applied after each convolution. Although we defined the input x as a vector for notational clarity, the actual input to CocoNet is a three dimensional tensor $x \in \{0,1\}^{V \times T \times P}$, where V is the fixed number of voices, T is the number of time-steps, P is the number of pitches, and each voice is a piano roll. As a result, $D = VTP$ and O is a random ordering of tensor indices rather than vector indices.

In order to evaluate the model, the authors conduct a listening test, asking participants to select the more “Bach-like" musical excerpt given a pair of samples. The experimental results demonstrate that autoregressive sampling (sampling $p(x_{O_j}|x_{O_{<j}})$ for $0 < j < D$) generates lower quality samples than an annealed independent blocked Gibbs (iGibbs) sampling procedure. Interestingly, although the mean rating for the actual Bach chorales was higher, there was not a significant difference between actual Bach chorales and those generated using iGibbs.

Polyphony with Multivariate Vectors

Lattner et al. (C2) apply constraints while sampling from a convolutional restricted Boltzmann machine (C-RBM), to enforce structural similarity to a template piece $\mathcal{T}$. Concretely, they specify three constraints on the generated piano roll $\mathcal{G} \in \{0,1\}^{T \times P}$: a self-similarity constraint, which minimizes the mean squared error (MSE) between self-similarity matrices derived from $\mathcal{T}$ and $\mathcal{G}$; a tonality constraint, which minimizes the MSE between key estimation matrices derived from $\mathcal{T}$ and $\mathcal{G}$; and a meter constraint, which minimizes the MSE between onset distribution vectors derived from $\mathcal{T}$ and $\mathcal{G}$. Samples are drawn from the model by alternating between two steps: a gradient descent step, to minimize MSE for the constraints; and a Gibbs sampling step, to sample from the learned data distribution. The authors provide examples generated by a model trained on 3 Mozart sonatas to validate their approach.

2.4.3 Generative Adversarial Network

Generative Adversarial Nets (GAN) [34] are an unsupervised training framework for generative models. A GAN is comprised of two networks, a discriminator D and a generator G, which compete in a minimax two-player game. D attempts to determine whether a sample belongs to the generative distribution or the training data distribution, while G attempts to generate samples that D will miss-classify. Concretely, G accepts a random vector z as input, and produces an output with the same dimensionality as the training data, while D accepts a sample and outputs a scalar value on the range $[0,1]$ corresponding to the probability that the sample is real. Notably, there are no restrictions on the exact architecture of D and G. A variety of modifications to the loss function have been proposed [44] in an effort to stabilize training and promote quicker convergence.

Polyphony with Multivariate Vectors

Dong et al. use the GAN framework to train MuseGAN (E1), a system that generates multi-track piano rolls $x \in \{0,1\}^{M \times T \times P}$, where M is the number of tracks. They experiment with three different architectures. The Jamming model is comprised of M generators, and M discriminators. Each generator is conditioned on an independent random vector, and evaluated by the corresponding discriminator. The Composer model consists of a single generator G, which, given a single random vector z, generates M tracks that are evaluated by a single discriminator D. The Hybrid model, which is comprised of M distinct generators that accept an *inter-track* random vector (z) and an *intra-track* random vector (z_i), generate music evaluated by a single discriminator. Since a single measure is generated at a time, the authors propose two methods to generate a sequence of measures. The first, involves a generator G_{temp} that maps a random vector to a sequence of latent vectors, from which a sequence of measures can be generated. The second, involves conditioning the generation on previously generated measures.

The models are trained using a subset of the Lakh Midi Dataset (DT7) containing only Rock songs in a 4/4 time-signature. Several metrics are used to dissect the differences between models: the percentage of empty bars; the number of distinct pitch classes used in a bar; the percentage of notes longer than a 32^{nd} note the percentage of notes in a 8 or 16 note beat pattern; and the tonal distance between tracks. A listening test demonstrated that the hybrid model was preferred by participants, in a comparison between the Jamming, Composer and Hybrid models.

Polyphony with Multivariate Sequences

Mogren proposed the Continuous RNN GAN (C-RNN-GAN) (E2), which models each note with four real valued scalars: note length, frequency, velocity and time-delta in seconds. The model is trained on 3697 MIDI files scraped from the internet, featuring 160 different classical composers. The authors compare the performance of their model against a baseline, an RNN trained with a maximum likelihood criterion, with respect to several musical metrics. The metrics measure scale consistency, the range of generated notes, the number of unique tones, the range of generated velocities, the amount of polyphony, and the number of 3-tone repetitions.

Although the adversarially trained model outperforms the maximum likelihood RNN with respect to the metrics, musical excerpts generated by the C-RNN-GAN appear to be very low-quality. Previous research has shown that log-likelihood and sample quality are largely independent for GANs that generate high-dimensional image outputs [89], demonstrating that evaluation metrics must be validated against human perception, which was not the case in the experiment evaluating the C-RNN-GAN.

2.4.4 Recurrent Neural Networks

Recurrent neural networks (RNN) [86] are used to model sequential data. They have an internal memory state which allows the network to make predictions based on previously seen data. More concretely, given a sequence $x = \{x_0, x_1, ..., x_n\}$, an RNN is trained to model $p(x_t|x_{<t})$. In order to address the vanishing gradient issue, where the learning signal becomes vanishingly small after being back-propagated over many time-steps, more complex recurrent units have been proposed. These include the Long Short-Term Memory (LSTM) [43] and the Gated Recurrent Unit (GRU) [16].

Fixed Voice Polyphony with Multivariate Sequences

Hadjeres et al. propose DeepBach (F1), a network on the Bach chorales dataset (DT2). The data is represented as a sequence of 6-dimensional vectors representing each time-step at a sixteenth note resolution. The first 4 dimensions of the input specify the soprano, alto, tenor and bass pitches respectively. A special hold token is used to represent ties. The last two dimensions specify the time relative to a beat, and whether or not the time-step is a fermata.

The architecture has four components: the *past* encoder, a 2-layer LSTM that summarizes the last T steps; the *future* encoder, a 2-layer LSTM that summarizes the next T steps in reverse order; the *present* encoder, a feed forward network that encodes information about the current time-step; and a feedforward network that predicts a single pitch based on the three encoded vectors. For both the *past* and *future* encoder, only the last output is kept, which encodes information from the entire sequence passed into the encoder. For their experiments, $T = 16$, which means the network only considers the bar before and after the notes being predicted. A separate model is trained for each voice (Soprano, Alto, Tenor, Bass). The authors describe a Pseudo-Gibbs sampling algorithm which is used to generate new pieces.

To evaluate their model, the authors conducted a listening test asking participants to discriminate between a generated piece and the an actual Bach chorale. The results generally indicated that participants had a difficult time discriminating between DeepBach's outputs and actual Bach chorales.

Homophony with Multivariate Sequences

Joslyn et al. (F2) generate music via hashing. The system consists of two LSTM networks, a forward m_f and backward m_b encoder, both of which produce k-ary codes (a vector of integers) from a single measure of musical material. A multivariate sequence representation is used, where each time-step is a 29-dimensional binary vector that indicates if the melody is articulated, the octave of the melody, the melody pitch class, and the chord type. Given a corpus $C = \{C^1, ..., C^N\}$, where each piece C^i is represented as a sequence of measures $C^i = \{C^i_1, C^i_2, ..., C^i_{n_i}\}$, the network is trained to minimize $|m_f(C^i_t) - m_b(C^i_{t+1})|$ for all valid i and t. Once the encoders are trained, the forward and backward k-ary codes for all measures added to a database. Then, to generate a continuation for a measure x, the database is queried to find a measure y for which $|m_f(x) - m_b(y)|$ is minimized or below some threshold. The authors perform an informal evaluation of the samples generated by a model trained on the Nottingham corpus (DT4).

Polyphony with Multivariate Sequences

Boulanger et al. (F7) generate polyphonic music by extending an RNN with a distribution estimator. Here, the network learns to predict a column of a piano roll, representing all the notes sounding at a particular time-step, conditioned on previous columns. In order to model the high-dimensional multivariate output (a column of a piano roll) at each time-step, the outputs of the RNN are used to condition a distribution estimator, which is then used to predict the final output. The authors experiment with a Neural Autoregressive Density Estimator (NADE) and an Restricted Boltzmann Machine (RBM). Experimental results demonstrate that the RNN-NADE is more effective at modeling music than the RNN-RBM, as it achieves the lowest log-loss on a collection of datasets (Bach chorales (DT2), Piano-Midi.de (DT6), MuseData (DT5), and Nottingham (DT4)). This architecture was extended to create two systems [32, 95], both using a Deep Belief network (DBN) as the density estimator. Although the log-loss across the same test datasets is lowest using the LSTM-DBN [95], it should be noted that the likelihoods are optimistic bounds on the intractable true values [96]. Furthermore, no listening tests were conducted to validate any of these models.

Johnson (F4) proposes two approaches to generating polyphonic music. The Tied-Parallel LSTM NADE (TP-LSTM-NADE) consists of 128 note-networks (TP-LSTM), each of which are responsible for a single pitch $p \in [0, 128)$, and share weights with all other note-networks. Let $x \in \{0,1\}^{T \times P}$ be a piano roll with $P = 128$. For a note p at time-step t, each note-network is provided with three inputs: a local window $w_{t-1} = [x_{t-1,i} : p - 12 \leq i \leq p + 12]$, containing all notes in the previous time-step within a range of ± 12 semitones; a relative pitch class vector v_{t-1}, where $v^i_{t-1} = \sum_{m=-\infty}^{\infty} x_{t-1,p+i+12m}$, counting the number of pitch classes that are played at each interval to relative to p; and the midi pitch number p. The TP-LSTM outputs at time-step t, are simply the concatenated output for each $p \in [0, 128)$, which can be accomplished in practice through batch processing as the network weights are tied across each note-network. The outputs of the TP-LSTM are used to condition a NADE, based on the approach used by the RNN-NADE (F7). The Bi-Axial LSTM (BALSTM), replaces the NADE portion of the TP-LSTM-NADE with an LSTM, which we refer to as the chord-LSTM. In contrast to the TP-LSTM that has recurrent connections along the time axis, the chord-LSTM has recurrent connections along the note axis. The chord-LSTM is provided with two inputs: the final outputs from the TP-LSTM, and the previous time-step x_{t-1}. To evaluate the TP-LSTM-NADE and the BALSTM, the authors use the same datasets as Boulanger et al., demonstrating an improvement over the RNN-NADE. They also perform an informal qualitative analysis.

Mao et al. propose DeepJ (F3), an extension of the BALSTM architecture, where the chord-LSTM produces three outputs: the probability of an onset, the probability of a tie, and the velocity of the note scaled onto the range $[0, 1]$. In addition, they condition the network using a n-dimensional style vector s. The authors train a model on baroque, romantic and classical music, and conduct two listening tests. The first compared samples generated by the BALSTM and DeepJ, finding that listeners preferred those generated by DeepJ. This result is not surprising, as the BALSTM does not generate velocity. Another experiment compared the listeners genre (baroque, romantic and classical) classification accuracy for human-composed and computer-generated samples. A statistical test revealed no significant difference between either condition (human-composed and computer-generated). For both conditions the listeners accuracy ranged from 50-60%. More than anything, this seems to indicate that more experienced listeners should have been selected.

Fixed Voice Polyphony with Univariate Sequences

BachBot (F8) is a system that generates Bach chorales using an LSTM network trained on a univariate representation of the Bach Chorale dataset (DT2). Musical material is represented using 130 distinct tokens, 128 corresponding to each possible midi-note, a token to indicate a tie, and a token to indicate a time-shift of one sixteenth note. As a result, each time-step can be represented by 9 tokens, 4 tokens for the pitches in each voice, 4 tokens to indicate whether or not the notes are tied, and one time-shift token. The listening experiment, which collected over 2,300 participants, is the largest study of its kind. Each participant was asked to discriminate between a computer-generated chorale and an actual Bach chorale. For some versions of the model, they found that participants were unable to reliably discriminate the generated music, making a compelling case for the quality of the generated material.

Homophony with Univariate Sequences

Using a text-based univariate representation, Choi et al. (F9) model jazz chord sequences with a 2-layer LSTM. Chord sequences are represented in text format (e.g. C:maj | F:maj G:maj | D:min7), where | denotes a bar-line. As a result, the representation consists of 64 distinct tokens, including 52 tokens for lowercase and uppercase letters, 10 tokens for numbers, and 2 additional tokens (|,:). The authors conduct an informal analysis of the generated material.

Polyphony with Univariate Sequences

Walder et al. (F12) propose a method to predict the pitch content given a fixed rhythmic structure. Note that this system uses a multivariate sequence as input, and a univariate sequence as output. Since the system is modeling a univariate sequence, and exhibits similarities to other systems in this section, we include it here. The network attempts to predict the next pitch given: a boolean vector $x \in \{0,1\}^{128}$ specifying each pitch that is currently sounding, and five scalar values describing the rhythmic structure $(t, \Delta t_{\text{event}}, \Delta t_{\text{step}}, \epsilon_{\text{on}}, \epsilon_{\text{off}})$. t indicates the relative position within the piece scaled onto the range $[0,1]$. Δt_{event} denotes the duration of the current note being predicted. Δt_{step} is the time shift since the last prediction. Both Δt_{event} and Δt_{step} are scaled relative to a quarter note (e.g. a dotted quarter note is 1.5). ϵ_{on} is 1 only if notes are being turned on at the same time as the last prediction, and ϵ_{off} is 1 only if notes are being turned off at the same time as the last prediction. One benefit of this representation, is that it does not require uniform time-steps, as Δt_{step} can take on any scalar value. Since notes are sorted lexicographically, first by onset and then by pitch, notes lower than or equal to the last predicted note can be masked when $\Delta t_{\text{step}} = 0$. The authors train a multi-layer LSTM, based on the architecture used by Zaremba et al. [102], on the four datasets used by Boulanger et al. (F7) (DT2, DT6, DT5, DT4), and compare log-loss against several models. They also train the network on a larger dataset, combining the four aforementioned datasets, and the Classical Music Archives dataset, informally analyzing the generated samples.

PerformanceRNN (F11) is an LSTM network trained on a univariate representation to generate expressive polyphonic music. The representation consists of 128 `NOTE_ON` tokens, 128 `NOTE_OFF` tokens, 125 `TIME_SHIFT` tokens corresponding to each 8ms increment in the range [8ms,1s], and 32 `VELOCITY` tokens, which apply to all subsequent note onsets. Informal feedback from musicians is used to validate the quality of the generated samples. Meade et al. (F10) experiment with various approaches to conditioning generation. They train PerformanceRNNs conditioned on: composers; styles; time-periods; composers clustered by latitude, longitude and birth year; scale type (major, minor and unknown); tempo; form (e.g. ballade); and velocity. Overall, the authors found the results unsatisfactory, as the conditioning fails exert enough control over the sampling process.

2.4.5 Transformer

The Transformer [92] is a deep feedforward network that makes use of soft attention, which enables a network to focus on a specific subset of its inputs. The implementation presented in the original paper consists of 6 encoding layers and 6 decoding layers. Each encoding layer consists of two parts: a self-attention sub-layer, and a feedforward sub-layer. The self-attention sub-layer is multi-headed, which means attention is computed across the input n times. Furthermore, attention is computed with a position matrix S, specifying the relative position of each input. Each decoding layer includes an additional attention-mechanism, placed in between the self-attention sub-layer and the feedforward sub-layer, which computes attention across the outputs of the final encoding layer. In contrast to RNNs which transmit information about past time-steps via the hidden state, an attention-based network can directly attend to time-steps in the distant past, which improves the learning of long-term dependencies.

Polyphony with Univariate Sequences

The Music Transformer (G2) is based on the Transformer architecture. The main modification that the authors propose, is a memory efficient method for computing the relative position matrix S. Due to memory limitations, it is not possible to take an entire piece as input, so the model is trained on randomly selected excerpts of length $L = 2048$ tokens. The authors use the same representation as the PerformanceRNN (F11) to encode the Maestro dataset (DT10). They also train on the Bach chorales dataset (DT2), lexicographically sorting notes first by onset and then by pitch, and using 128 tokens to represent the pitches.

To evaluate the Music Transformer, the authors compare log-loss on held-out test data against a baseline model. They compare the system against CoCoNet (C1) for the Bach chorales corpus, and against the PerformanceRNN (F11) for the Maestro corpus. In both cases, the Music Transformer significantly outperforms the baseline. The authors conduct a listening test for the Maestro model, which demonstrates that the Music Transformer generates higher quality output than the PerformanceRNN. Qualitative experimentation suggests that the model is able to return to previous motifs, demonstrating some knowledge of long term structure.

MuseNet (G1) is a 72-layer Transformer architecture with 24-headed attention over a context of $L = 4096$ tokens. Their dataset is sourced from the internet, and includes Classical Archives, files scraped from BitMidi, and the Maestro dataset (DT10). Musical material is represented as a univariate sequence. After experimenting with different representations, Payne found that combining P pitches, N instruments and V velocities into a single token produced the best results, resulting in $P \times N \times V$ distinct tokens. Some additional tokens are used to condition generation according to a composer or an instrument combination. There is no formal evaluation of the model.

LakhNES (G5) uses the Transformer architecture to generate NES music, which is comprised of four monophonic tracks: melody, harmony, bass and drums. Musical material is represented as a univariate sequence, using a similar approach to MuseNet (G1), where there are distinct note onset tokens for each pitch on each track, and the note onset from all four tracks are interleaved into a single sequence. This results in $4P$ tokens, where P is the number of pitches represented. LakhNES is trained in two phases. In the first phase, a heuristic is used to transform MIDI files from the Lakh MIDI dataset (DT7) into 4 track NES style arrangements. For example, percussive MIDI instruments are mapped onto the drum track, while melodic MIDI instruments like the saxophone and flute are mapped onto the melody and harmony tracks. Polyphonic tracks are simply discarded. After the first, pre-training phase is complete, the network is fine-tuned using the much smaller NES Music Database (DT13). The authors evaluate their system using a listening test, which demonstrates that the transformer architecture outperforms LSTM and n-gram models (trained using the same procedure). However, the results show that the participants were still able to distinguish human composed NES music from music generated by LahkNES.

MusIAC (G4) is a generative system designed to accommodate infilling, using a fixed schema of three tracks: melody, bass, and harmony (homophonic). The system allows for a variety of attribute controls, including track-level control over note density, polyphony, and occupation (the inverse of rest percentage); as well as bar-level control over tensile strain and cloud diameter. Tensile strain and cloud diameter are parameters derived from the Spiral Array [14], which is a 3-dimensional harmonic space where individual pitches correspond to specific 3-dimensional coordinates. The system is trained using the Lakh MIDI Dataset (DT7). No listening test is conducted to evaluate the quality of the music generated by the system.

The Multi-Track Music Machine (MMM) (Ch. 7) is based on the Transformer architecture, and is designed to support co-creative music composition workflows. MMM supports the infilling of musical material on the track and bar level, and can condition generation on particular attributes including: instrument type, note density, polyphony level, and note duration. In order to integrate these features, we employ a different type of representation for musical material, creating a time-ordered sequence of musical events for each track and concatenating several tracks into a single sequence, rather than using a single time-ordered sequence where the musical events corresponding to different tracks are interleaved. A more detailed discussion of MMM can be found in Chapter 7.

2.4.6 Variational Autoencoder

A variational autoencoder (VAE) [54] is comprised of an encoder, which encodes an input x into a latent vector z, and a decoder, which decodes z into an output $\hat{x}$ with the same dimensionality as x. The network is trained to jointly minimize two objectives: the reconstructed output $\hat{x}$ should be as similar to x as possible, and the distribution of encoded samples in the latent space should be as close to a Normal distribution as possible. The importance assigned to each objective can have a significant impact on the samples produced by the decoder, as overemphasizing the latter objective impairs the networks ability to reconstruct a given input, while overemphasizing the former objective results in a disorganized latent space. An optimal VAE allows for interpolation between samples, and the generation of novel samples by providing a randomly sampled latent vector z to the decoder.

Polyphony with Multivariate Sequences

Brunner et al. (H1) use a VAE to manipulate the musical style of a single bar of music. In addition to a standard M-track piano roll $x \in \{0,1\}^{M \times T \times P}$, two other matrices are passed into the network. The instrument assignment matrix $I \in \{0,1\}^{M \times N}$, where N is the number of instruments, represents the instrument corresponding to each of the M tracks. The velocity matrix $V \in \mathbb{R}^{M \times T \times P}$, where velocity values are mapped onto the range $[0.5, 1]$.

There is a separate GRU encoder for x, I and V. Since RNNs can only process 2-dimensional input, the first two axes are combined resulting in matrices $\hat{x} \in \{0,1\}^{MT \times P}$ and $\hat{V} \in \{0,1\}^{MT \times P}$, so that each successive set of M vectors corresponds to a single time-step. The outputs of the GRU encoders are combined and passed through some feedforward layers to produce a latent vector z. $z_{<k}$ is passed to a style classifier, which is trained to predict the correct style, ensuring that the first k dimensions of the latent vector represent style information. There are three separate GRU decoders which aim to reconstruct $\hat{x}$, $\hat{V}$ and I given the latent code z.

To evaluate the model, the authors train an ensemble of style classifiers C which take $\hat{x}$, $\hat{V}$ and I as input. Using a dataset consisting of 5 different styles (Classic, Jazz, Pop, Bach and Mozart), models are trained on two styles ($k = 2$), and for each musical excerpt m another version $\hat{m}$ is generated where the style swapped. Then the difference between $C(m)$ and $C(\hat{m})$ is calculated to provide an estimate of how well the style was manipulated.

Fixed Voice Polyphony with Univariate and Multivariate Sequences

Roberts et al. propose MusicVAE (H2), a VAE with a bi-directional LSTM encoder, and a hierarchical LSTM decoder to model a variety of subsets of a trio consisting of melody, bass and drums.

The melody and bass are represented using a univariate sequence consisting of 128 `NOTE_ONSET` tokens, 1 `REST` token, and 1 `NOTE_OFF` token. The drums are represented using a multivariate sequence resembling a piano roll. In contrast to a standard LSTM, which has recurrent connections moving forward in time, a bi-directional LSTM also incorporates recurrent connections that move backwards in time. When generating particularly long sequences, the authors found that a standard LSTM decoder will tend to ignore the latent code. In order to address this issue, a *Conductor* LSTM generates a sequence of latent codes, one for each measure, taking the original latent code as input. Then a standard LSTM decoder generates each measure using the latent codes produced by the *Conductor*. Models are trained to generate 2 bars of melody, 2 bars of drums, 16 bars of melody, 16 bars of drums, and 16 bars of the full trio. For the trio model, three separate encoders (melody, bass, drums) produce a latent vector, which is used to condition the melody, bass and drum decoders. The authors conduct a listening test, which demonstrates that the hierarchical decoder produces higher quality samples than a non-hierarchical decoder, and that the generated samples are of similar quality to the original data.

2.5 Variable Neighborhood Search

Herremans et al. frame the music generation problem as an optimization problem, developing Morpheous (I1), a system that generates pitches for a fixed rhythmic structure, with repetition constrained by a template piece. An optimal solution matches a harmonic tension profile provided by a user. Tension is computed using the Spiral Array [14], a 3-dimensional harmonic space where each pitch corresponds to a specific 3-dimensional coordinate. To measure the tension of a musical excerpt, the excerpt is divided into $1/8$ note segments. For each segment, three metrics are calculated: *Cloud Diameter*, which measures the dispersion of notes within a segment; *Cloud Momentum*, which measures the amount of tonal movement between successive segments; and *Tensile strain*, which measures the distance between each segment and the global key. Repetition in pitch/time space is detected using the COSIATEC and SIATECCompress algorithm [65]. Then, Variable Neighborhood Search is used to find the optimal pitch assignments, given the repetition structure, and the tension constraints. Unfortunately, in some circumstances, the optimization process ends up converging to the original piece. To evaluate the system, the authors informally discuss some of the material that was generated.

2.6 Challenges and Opportunities

Based on the limitations of CSPG systems discussed in our survey, we propose five challenge areas: the quality of data; the generation of non-Western music; explicit control over the generation process; comparative evaluation of CSPG systems; and the development of increasingly creative CSPG systems.

2.6.1 Data Quality

There are several issues with regards to data quality. The first, is related to the loudness of individual notes, which is referred to as velocity in the MIDI protocol [67]. In many cases, meaningful velocity information is not present in MIDI files, as all velocity values may be set to the same value, or only a small range of the possible velocity values are used. This makes it difficult to train a CSPG system that models composition and performance jointly, such as the PerformanceRNN (F11). Currently, the largest dataset containing valid velocity data is the Piano-e-competition dataset, which is approximately 100 times smaller than the Lakh Midi Dataset [81]. Another issue, is the lack of reliable voice separation. For example, the musicVAE (H2), which generates musical trios consisting of a bass-line, melody and drums, was trained using a heuristic to infer which MIDI tracks belonged to a particular voice type (i.e. melody). In cases where a MIDI file contains more than one candidate melody or bass-line, all possible combinations are used to train the model. Unfortunately, some combinations will lack essential aspects of the original piece, diluting the quality of the samples that the network is trained on.

2.6.2 Generating non-Western Music

With relatively few exceptions, data representations and computational models are designed to accommodate Western music, which is broadly characterized by its use of 12 pitch classes per octave, and simple meter. As a result, musical traditions which make use of microtonal pitches, such as Indian, Persian, and Indonesian Gamelan music [10], are severely underserved. However, digital representation protocols are currently under development for traditional Indian music [61]. Notably, the C-RNN-GAN (E2) is capable of handling microtonal pitches, however, the authors limit their experiments to Western music. With respect to this challenge, the development of accessible datasets should be a focus for future research, followed by research examining alternate data representations and computational models.

2.6.3 Controlling Generation

In many contexts, it is desirable to have some degree of control over the generation process, however, many models do not afford any type of control. There are different degrees of control ranging from low-level control, which specifies the explicit constraints, to high-level control, which conditions generation on general attributes.

Infilling, an example of low level control, conditions generation on a set of notes, requiring the system to predict the remaining notes. Although Liang et al. (F8) infill a single chorale voice by restricting the sampling process, the model was unable to anticipate these constraints, resulting in a decrease in sample quality. The anticipationRNN [37] addresses this problem, using a bi-directional RNN encoder to learn constraint embeddings which condition another RNN at each time-step, however, this approach has only been applied to monophonic music. CoCoNet (C1) is trained under the

NADE framework, which is designed to accommodate infilling, however, this model only handles fixed-voice polyphony.

In contrast, specifying structural repetition within a piece is an example of mid-level control. Three models have been proposed that constrain structure in this manner (I1, A6, C2), however there are limitations to each model. The C-RBM (C2) only works on small datasets (∼5 pieces), as the authors explicitly state they rely on overfitting, requiring a new model to be trained for each small dataset. Furthermore, inference is very slow. MorpheuS (I1) tends to converge to the template piece from which the structure is extracted, and Racchmaninof (A6) exhibits poor performance on corpora that are more complex than Bach chorales, due to the limited capacity of first-order Markov model. Factorized generation could also be considered mid-level control. Examples of factorized generation include generating pitch conditioned on rhythm and vice versa. Walder et al. (F12) train a model to generate pitch given rhythm, using a complex representation. Yang et al. [101, 100] describe two approaches to the factorized (pitch and rhythm) generation of melodies.

Another approach to control, which has been explored in several systems (F10, H1, G1), involves conditioning generation on high-level attributes, such as style, tempo, and instrumentation. MuseNet (G1), a Transformer-based neural network, provides an example of this type of control, allowing the user to mix and match composers with a variety of instrument combinations.

Future work should focus on integrating these different levels of control into CSPG systems, allowing for generation to be steered by the user. Extending the style manipulation model proposed by Bruner et al. may prove fruitful, as the choice to train the model on only two styles may have limited the models ability to learn what constitutes a single style. Further experimentation with attention based networks, such as the Transformer (G2, G1), is worth exploring, as their ability to learn long-term musical dependencies is currently unparalleled. Furthermore, the attention mechanism could be adapted to allow for the anticipation of future constraints, building on the ideas motivating the AnticipationRNN architecture. Template-based approaches such as Racchmaninof (A6) and MorpheuS (I1) could be implemented using deep learning, as the AnticipationRNN should be capable of imposing constraints on an arbitrary univariate sequence, despite only being tested on a univariate representation of melody.

2.6.4 Plagiarism, Novelty, and Other Issues Related to Creativity

Although CSPG systems are trained in an attempt to emulate the stylistic characteristics of a particular corpus $\mathcal{C}$, these systems may in fact plagiarize directly from $\mathcal{C}$ in practice. Since CSPG systems are usually trained to predict patterns found in $\mathcal{C}$, not patterns that simply exhibit the same stylistic characteristics as those found in $\mathcal{C}$, it is unsurprising that plagiarism can be an issue. At worst, near-exact replicas are generated (I1), however, frequent smaller instances of plagiarism do demonstrate a decreased capacity for generating novel musical material, and should be minimized. Potential solutions include: training models on a superset of $\mathcal{C}$ produced via novel data augmentation techniques; and sampling with constraints prohibiting plagiarism, an approach which has been applied to melodies [74]. In addition, the development of methods that calculate the degree to which a system plagiarizes would be beneficial.

Since most systems exhibit *Exploratory* creativity [7], generating musical material within a particular style, future research may focus on systems that exhibit *Combinatorial*, and where possible, *Transformational* creativity. Systems have been proposed which manipulate style (H1, F10, G1), but this area of research is still in its infancy. Factorized generation presents interesting opportunities for developing systems that exhibit *Combinatorial* creativity, allowing for different factors to be generated in different styles. Adapting the Creative Adversarial Network [29], a GAN-based network that attempts to generate images in novel style, to the music domain could prove interesting.

2.6.5 Evaluation of CSPG Systems

There are two main challenges surrounding the evaluation of generative models. First of all, the number of generated samples that are included in a listening test is still quite small, increasing chances that the system will be misrepresented. For example, the BachBot (F8) and Music Transformer (G2) listening experiments were comprised of 36 and 10 generated excerpts, respectively. To address this issue, some studies include analytic experimental results with a much larger sample size (E1, A2, B3), however, this approach is uncommon. Note that this topic is addressed in greater detail in Chapter 7. Second, the lack of a standardized evaluation procedure makes comparison difficult. This difficulty is compounded for listening studies, which would have to be completely redone to compare additional models. Some analytic approaches have been proposed [48, 99], but are not yet widely adopted. Although code is increasingly being open sourced, the author-trained model is rarely provided, which makes a large scale analysis of CSPG systems difficult. Future research should focus on standard practices and methodologies for system evaluation that can be easily adopted by the research community.

2.7 Conclusion

Motivated by practical use-cases in the creative industries, and recent developments related to A.I., a variety of CSPG systems have been proposed in recent years. In order to organize this large body of research, we provided a typology of CSPG systems, organizing each system according to the data representation, computational model, and generated musical texture. Overall the results are encouraging, as state-of-the-art systems are capable of generating human competitive musical artifacts, which are nearly indistinguishable from human-composed music. However, there still challenges which have yet to be sufficiently addressed, ranging from data quality to the evaluation process, which are undoubtedly critical aspects of future research.

Bibliography

[1] Kat Agres, Jamie Forth, and Geraint A Wiggins. "Evaluation of musical creativity and musical metacreation systems". In: *Computers in Entertainment (CIE)* 14.3 (2016), pp. 1–33.

[2] Moray Allan and Christopher Williams. "Harmonising chorales by probabilistic inference". In: *Advances in neural information processing systems*. 2005, pp. 25–32.

[3] Teresa M Amabile. "Social psychology of creativity: A consensual assessment technique." In: *Journal of personality and social psychology* 43.5 (1982), pp. 997–1013.

[4] Christopher Ariza. "The interrogator as critic: The turing test and the evaluation of generative music systems". In: *Computer Music Journal* 33.2 (2009), pp. 48–70.

[5] Leonard E Baum and Ted Petrie. "Statistical inference for probabilistic functions of finite state Markov chains". In: *The annals of mathematical statistics* 37.6 (1966), pp. 1554–1563.

[6] Bruce Benward and Marilyn Saker. *Music in Theory and Practice*. Music in Theory and Practice v. 1. McGraw-Hill, 2003.

[7] Margaret A Boden. *The creative mind: Myths and mechanisms*. Routledge, 2004.

[8] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription". In: *International Conference on Machine Learning* (2012).

[9] Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. "MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer". In: *Proceedings of the 19th International Symposium for Music Information Retrieval*. 2018, pp. 747–754.

[10] Edward M Burns. "Intervals, scales, and tuning". In: *The psychology of music*. Elsevier, 1999, pp. 215–264.

[11] Carlos Eduardo Cancino-Chacón, Maarten Grachten, Werner Goebl, and Gerhard Widmer. "Computational models of expressive music performance: A comprehensive and critical review". In: *Frontiers in Digital Humanities* 5 (2018), p. 25.

[12] Augustin Cauchy. "Méthode générale pour la résolution des systemes d'équations simultanées". In: *Comp. Rend. Sci. Paris* 25.1847 (1847), pp. 536–538.

[13] Gong Chen. “Who Composes the Music?: Musicality Evaluation for Algorithmic Composition via Electroencephalography”. In: *Proceedings of the 2017 ACM on Multimedia Conference MM 2017 Mountain View CA USA October 23-27 2017*. 2017, pp. 826–830.

[14] Elaine Chew. *Mathematical and Computational Modeling of Tonality - Theory and Applications*. Springer, 2014.

[15] Keunwoo Choi, George Fazekas, and Mark Sandler. “Text-based LSTM networks for automatic music composition”. In: *arXiv preprint arXiv:1604.05358* (2016).

[16] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: *NIPS 2014 Deep Learning and Representation Learning Workshop* (2014).

[17] Tom Collins and Robin Laney. “Computer-generated stylistic compositions with long-term repetitive and phrasal structure”. In: *Journal of Creative Music Systems* 1.2 (2017).

[18] Tom Collins, Jeremy Thurlow, Robin Laney, Alistair Willis, and Paul Garthwaite. “A comparative evaluation of algorithms for discovering translational patterns in baroque keyboard works”. In: *Proceedings of the International Symposium on Music Information Retrieval* (2010), pp. 3–8.

[19] Simon Colton, John William Charnley, and Alison Pease. “Computational Creativity Theory: The FACE and IDEA Descriptive Models.” In: *ICCC*. Mexico City. 2011, pp. 90–95.

[20] Association for Computational Creativity. *Computational Creativity*. [Online; accessed 17-12-2019]. 2019.

[21] Darrell Conklin and Ian H Witten. “Multiple viewpoint systems for music prediction”. In: *Journal of New Music Research* 24.1 (1995), pp. 51–73.

[22] Michael Scott Cuthbert and Christopher Ariza. “music21: A toolkit for computer-aided musicology and symbolic music data”. In: *Proceedings of the 11th International Society for Music Information Retrieval Conference* (2010), pp. 637–642.

[23] Roger B Dannenberg. “Music representation issues, techniques, and systems”. In: *Computer Music Journal* 17.3 (1993), pp. 20–30.

[24] Mark DeVoto. *Encyclopædia Britannica*. Sept. 2017.

[25] Chris Donahue, Huanru Henry Mao, Yiting Ethan Li, Garrison W Cottrell, and Julian McAuley. “LakhNES: Improving multi-instrumental music generation with cross-domain pre-training”. In: *Proc. of the 20th International Society for Music Information Retrieval Conference*. 2019, pp. 685–692.

[26] Chris Donahue, Huanru Henry Mao, and Julian McAuley. “The NES music database: A multi-instrumental dataset with expressive performance attributes”. In: *arXiv preprint arXiv:1806.04278* (2018).

[27] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018, pp. 34–41.

[28] Arne Eigenfeldt and Philippe Pasquier. “Realtime generation of harmonic progressions using controlled markov selection”. In: *Proceedings of the International Conference on Computational Creativity*. 2010, pp. 16–25.

[29] Ahmed M. Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. “CAN: Creative Adversarial Networks, Generating "Art" by Learning About Styles and Deviating from Style Norms”. In: *Proceedings of the Eighth International Conference on Computational Creativity*. 2017, pp. 96–103.

[30] Kunihiko Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological cybernetics* 36.4 (1980), pp. 193–202.

[31] Paul A Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[32] Kratarth Goel, Raunaq Vohra, and Jajati Sahoo. “Polyphonic music generation by modeling temporal dependencies using a rnn-dbn”. In: *International Conference on Artificial Neural Networks*. Springer. 2014, pp. 217–224.

[33] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[35] Crina Grosan and Ajith Abraham. “Rule-Based Expert Systems”. In: *Intelligent Systems: A Modern Approach*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 149–185.

[36] Rui Guo, Ivor Simpson, Chris Kiefer, Thor Magnusson, and Dorien Herremans. “MusIAC: An extensible generative framework for Music Infilling Applications with multi-level Control”. In: *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*. Springer. 2022, pp. 341–356.

[37] Gaëtan Hadjeres and Frank Nielsen. “Anticipation-RNN: enforcing unary constraints in sequence generation, with application to interactive music generation”. In: *Neural Computing and Applications* (Nov. 2018).

[38] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. “Deepbach: a steerable model for bach chorales generation”. In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 1362–1371.

[39] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse H. Engel, and Douglas Eck. “Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset”. In: *7th International Conference on Learning Representations*. 2019.

[40] Jim Heckroth. “A tutorial on MIDI and wavetable music synthesis”. In: *Application Note, CRYSTAL a division of CIRRUS LOGIC* (1998).

[41] Dorien Herremans and Elaine Chew. “MorpheuS: generating structured music with constrained patterns and tension”. In: *IEEE Transactions on Affective Computing* (2017).

[42] Hermann Hild, Johannes Feulner, and Wolfram Menzel. “HARMONET: A neural net for harmonizing chorales in the style of JS Bach”. In: *Advances in neural information processing systems*. 1992, pp. 267–274.

[43] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[44] Yongjun Hong, Uiwon Hwang, Jaeyoon Yoo, and Sungroh Yoon. “How Generative Adversarial Networks and Their Variants Work: An Overview”. In: *ACM Computing Surveys* 52.1 (2019), 1:43.

[45] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.

[46] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron C. Courville, and Douglas Eck. “Counterpoint by Convolution”. In: *Proceedings of the 18th International Society for Music Information*. 2017, pp. 211–218.

[47] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. “Music Transformer: Generating Music with Long-Term Structure”. In: *7th International Conference on Learning Representations*. 2019.

[48] **Jeff Ens** and Philippe Pasquier. “CAEMSI: A Cross-Domain Analytic Evaluation Methodology for Style Imitation.” In: *Proceedings of the International Conference on Computational Creativity*. 2018, pp. 64–71.

[49] **Jeff Ens** and Philippe Pasquier. “Quantifying Musical Style: Ranking Symbolic Music based on Similarity to a Style”. In: *Proc. of the International Symposium on Music Information Retrieval*. 2019, pp. 870–877.

[50] Daniel D Johnson. “Generating polyphonic music using tied parallel networks”. In: *International conference on evolutionary and biologically inspired music and art*. Springer. 2017, pp. 128–143.

[51] Anna Jordanous. “Stepping Back to Progress Forwards: Setting Standards for Meta-Evaluation of Computational Creativity”. In: *Proceedings of the Fifth International Conference on Computational Creativity Ljubljana Slovenia June 10-13 2014*. 2014, pp. 129–136.

[52] Kevin Joslyn, Naifan Zhuang, and Kien A Hua. “Deep Segment Hash Learning for Music Generation”. In: *arXiv preprint arXiv:1805.12176* (2018).

[53] Maximos A. Kaliakatsos-Papakostas and Emilios Cambouropoulos. “Probabilistic harmonization with fixed intermediate chord constraints”. In: *Proceedings of the 40th International Computer Music Conference*. 2014, pp. 1083–1090.

[54] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *2nd International Conference on Learning Representations*. 2014.

[55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[56] Carolyn Lamb, Daniel G Brown, and Charles LA Clarke. “Evaluating computational creativity: An interdisciplinary tutorial”. In: *ACM Computing Surveys* 51.2 (2018), pp. 1–34.

[57] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. “Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints”. In: *Journal of Creative Music Systems* (2018).

[58] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[59] Hong-Ru Lee and Jyh-Shing R. Jang. “i-Ring: A system for humming transcription and chord generation”. In: *International Conference on Multimedia and Expo*. Vol. 2. IEEE. 2004, pp. 1031–1034.

[60] Feynman Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. “Automatic Stylistic Composition of Bach Chorales with Deep LSTM.” In: *Proceedings of the International Symposium on Music Information Retrieval*. 2017, pp. 449–456.

[61] Stanly Mammen, Ilango Krishnamurthi, A Jalaja Varma, and G Sujatha. “iSargam: music notation representation for Indian Carnatic music”. In: *EURASIP Journal on Audio, Speech, and Music Processing* 2016.1 (2016), pp. 1–12.

[62] Huanru Henry Mao, Taylor Shin, and Garrison W. Cottrell. “DeepJ: Style-Specific Music Generation”. In: *12th IEEE International Conference on Semantic Computing*. 2018, pp. 377–382.

[63] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[64] Nicholas Meade, Nicholas Barreyre, Scott C. Lowe, and Sageev Oore. “Exploring Conditioning for Generative Music Systems with Human-Interpretable Controls”. In: *Proceedings of the International Conference for Computational Creativity* (2019), pp. 148–155.

[65] David Meredith. “COSIATEC and SIATECCompress: Pattern discovery by geometric compression”. English. In: *Music Information Retrieval Evaluation eXchange (MIREX 2013)*. International Society for Music Information Retrieval, 2013.

[66] Olof Mogren. “C-RNN-GAN: Continuous recurrent neural networks with adversarial training”. In: *arXiv preprint arXiv:1611.09904* (2016).

[67] Robert A Moog. “MIDI: musical instrument digital interface”. In: *Journal of the Audio Engineering Society* 34.5 (1986), pp. 394–404.

[68] Ana-Maria Olteţeanu and Zoe Falomir. “comRAT-C: A computational compound Remote Associates Test solver based on language data and its comparison to human performance”. In: *Pattern Recognition Letters* 67 (2015), pp. 81–90.

[69] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. “This time with feeling: learning expressive musical performance”. In: *Neural Computing and Applications* (2018), pp. 1–13.

[70] François Pachet, Jeff Suzda, and Dani Martinez. “A Comprehensive Online Database of Machine-Readable Lead-Sheets for Jazz Standards.” In: *Proceedings of the International Symposium on Music information Retrieval*. 2013, pp. 275–280.

[71] Victor Padilla and Darrell Conklin. “Generation of Two-Voice Imitative Counterpoint from Statistical Models”. In: *IJIMAI* 5.3 (2018), pp. 22–33.

[72] Jean-François Paiement, Douglas Eck, and Samy Bengio. “Probabilistic melodic harmonization”. In: *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer. 2006, pp. 218–229.

[73] Alexandre Papadopoulos, Pierre Roy, and François Pachet. “Assisted lead sheet composition using flowcomposer”. In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2016, pp. 769–785.

[74] Alexandre Papadopoulos, Pierre Roy, and François Pachet. “Avoiding Plagiarism in Markov Sequence Generation”. In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014, pp. 2731–2737.

[75] Philippe Pasquier, Arne Eigenfeldt, Oliver Bown, and Shlomo Dubnov. “An introduction to musical metacreation”. In: *Computers in Entertainment (CIE)* 14.2 (2016), pp. 2–16.

[76] Christine Payne. “MuseNet”. In: *OpenAI* (Apr. 2019). openai.com/blog/musenet.

[77] Marcus T Pearce and Geraint A Wiggins. “Evaluating cognitive models of musical composition”. In: *Proceedings of the 4th international joint workshop on computational creativity*. Goldsmiths, University of London. 2007, pp. 73–80.

[78] Marcus T. Pearce, Darrell Conklin, and Geraint A. Wiggins. "Methods for Combining Statistical Models of Music". In: *Computer Music Modeling and Retrieval: Second International Symposium*. 2004, pp. 295–312.

[79] Dan Ponsford, Geraint Wiggins, and Chris Mellish. "Statistical learning of harmonic movement". In: *Journal of New Music Research* 28.2 (1999), pp. 150–177.

[80] Stanisław A Raczyński, Satoru Fukayama, and Emmanuel Vincent. "Melody harmonization with interpolated probabilistic models". In: *Journal of New Music Research* 42.3 (2013), pp. 223–235.

[81] Colin Raffel. "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching". PhD thesis. Columbia University, 2016.

[82] Mel Rhodes. "An analysis of creativity". In: *The Phi Delta Kappan* 42.7 (1961), pp. 305–310.

[83] Graeme Ritchie. "Some Empirical Criteria for Attributing Creativity to a Computer Program". In: *Minds and Machines* 17.1 (2007), pp. 67–99.

[84] Adam Roberts, Jesse H. Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music". In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 4361–4370.

[85] Pierre Roy and François Pachet. "Enforcing meter in finite-length markov sequences". In: *Twenty-Seventh AAAI Conference on Artificial Intelligence*. 2013, pp. 854–861.

[86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[87] Ian Simon, Dan Morris, and Sumit Basu. "MySong: automatic accompaniment generation for vocal melodies". In: *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM. 2008, pp. 725–734.

[88] Bob L. T. Sturm. *Going to use the Nottingham Music Database?* Oct. 2018.

[89] Lucas Theis, Aäron van den Oord, and Matthias Bethge. "A note on the evaluation of generative models". In: *Proceedings of the International Conference on Learning Representations*. 2016.

[90] Alan M. Turing. "Computing machinery and intelligence". In: *Parsing the Turing Test*. Springer, 2009, pp. 23–65.

[91] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. "Neural autoregressive distribution estimation". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 7184–7220.

[92] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[93] Dan Ventura. "Mere generation: Essential barometer or dated concept". In: *Proceedings of the Seventh International Conference on Computational Creativity*. Sony CSL, Paris. 2016, pp. 17–24.

[94] Andrew Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE transactions on Information Theory* 13.2 (1967), pp. 260–269.

[95] Raunaq Vohra, Kratarth Goel, and Jajati Sahoo. "Modeling temporal dependencies in data using a DBN-LSTM". In: *IEEE International Conference on Data Science and Advanced Analytics*. 2015, pp. 1–4.

[96] Christian Walder. "Modelling symbolic music: Beyond the piano roll". In: *Asian Conference on Machine Learning*. 2016, pp. 174–189.

[97] Raymond P Whorley and Darrell Conklin. "Music generation from statistical models of harmony". In: *Journal of New Music Research* 45.2 (2016), pp. 160–183.

[98] Gerhard Widmer and Asmir Tobudic. "Playing Mozart by analogy: Learning multi-level timing and dynamics strategies". In: *Journal of New Music Research* 32.3 (2003), pp. 259–268.

[99] Li-Chia Yang and Alexander Lerch. "On the evaluation of generative models in music". In: *Neural Computing and Applications* (2018), pp. 1–12.

[100] Ruihan Yang, Tianyao Chen, Yiyi Zhang, and Gus Xia. "Inspecting and Interacting with Meaningful Music Representations using VAE". In: *Proceedings of the International Conference on New Interfaces for Musical Expression* (2019).

[101] Ruihan Yang, Dingsu Wang, Ziyu Wang, Tianyao Chen, Junyan Jiang, and Gus Xia. "Deep Music Analogy Via Latent Representation Disentanglement". In: *Proceedings of the International Symposium for Music Information Retrieval* (2019). (in press).

[102] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization". In: *arXiv preprint arXiv:1409.2329* (2014).

Chapter 3

Building the MetaMIDI Dataset: Linking Symbolic and Audio Musical Data

As published in Ens, J. & Pasquier, P. (2021). *Building the MetaMIDI Dataset: Linking Symbolic and Audio Musical Data.* ISMIR.

Abstract

We introduce the MetaMIDI Dataset (MMD), a large scale collection of 436,631 MIDI files and metadata. MMD contains artist and title metadata for 221,504 MIDI files, and genre metadata for 143,868 MIDI files, collected during the web-scraping process. MIDI files in MMD were matched against a collection of 32,000,000 30-second audio clips retrieved from Spotify, resulting in over 10,796,557 audio-MIDI matches. In addition, we linked 600,142 Spotify tracks with 1,094,901 MusicBrainz recordings to produce a set of 168,032 MIDI files that are matched to the MusicBrainz database. We also provide a set of 53,496 MIDI files using audio-MIDI matches where the derived metadata on Spotify is a fuzzy match to the web-scraped metadata. These links augment many files in the dataset with the extensive metadata available via the Spotify API and the MusicBrainz database. We anticipate that this collection of data will be of great use to MIR researchers addressing a variety of research topics.

Keywords: Generative Music; Evaluation; Machine Learning; Big Data

3.1 Introduction

Large-scale metadata-rich MIDI datasets containing audio-MIDI matches [12, 10, 15] are indispensable in a wide variety of research contexts. For example, the Lakh Midi Dataset (LMD) [15] has been applied in many different contexts, including training generative music systems [7, 18], tempo-estimation [19], genre classification [9] and even as a primary data-source for new datasets [13, 8]. Motivated by the widespread demand for datasets of this nature, we created the MetaMIDI Dataset (MMD), which contains 2.4 times the number of MIDI files in the LMD, and audio-MIDI matches associating MIDI files with Spotify and MusicBrainz. To put the following numbers into context, we note that there is a many-to-one relationship between Spotify track ids and the actual audio recording. In this paper, we describe the process of assembling the dataset, which consists of the following contributions:

- Collection of 436,631 MIDI files.
- Scraped artist + title metadata for 221,504 MIDIs (10 times more than the LMD).
- Scraped genre metadata for 143,868 MIDIs.
- An improved audio-MIDI matching procedure, which produced 10,796,557 audio-MIDI matches linking 237,236 MIDIs to one or more tracks on Spotify.
- 829,728 high reliability audio-MIDI + scraped metadata (artist and title) matches linking 53,496 MIDIs to one or more tracks on Spotify.
- A method for linking Spotify tracks and MusicBrainz recordings, producing 8,263,482 unique links that associate 1,094,901 MusicBrainz recordings with 600,142 Spotify tracks.
- 168,032 MIDIs matched to MusicBrainz IDs via the Spotify/MusicBrainz linking procedure.

3.2 Data Collection

We scraped publicly available websites and were able to amass a collection of 436,631 unique MIDI files. Candidate websites were selected using a search engine to query various phrases including keywords such as MIDI, music, and a variety of musical genres. A list of the sites scraped and the number of MIDI files found on each site is provided in the dataset. Where possible, we also collected additional metadata, such as the artist, title and genre of associated with a particular MIDI file.

3.3 Audio Midi Matching

To augment MMD with additional metadata, we match the MIDI files against a large metadata-rich collection of audio clips. Although the LMD is comprised of audio-MIDI matches against the Million Song Dataset [1], we decided to use 30-second preview clips made available through the Spotify

API[1]. The primary motivation for this decision was the fact that the Spotify API provides over an order of magnitude more data. Using the Spotify API, we were able to collect 32,000,000 30-second MP3 files (over 13TB of raw data). To compute the audio-MIDI matches, we model our approach after the procedure employed by Raffel [15], who matched the 176,581 MIDI files in the LMD with 1,000,000 audio files in the Million Song Dataset [1]. However, we make some modifications to the matching algorithm to accommodate the large amount of data which was collected.

Raffel's audio-MIDI matching procedure is comprised of two stages [15]. In the first stage, which we refer to as the blocking stage, audio-MIDI pairs which are unlikely to be a match are removed from consideration. In the second stage, which we refer to as the matching stage, a confidence score (on the range [0,1]) is computed for each remaining audio-MIDI pair. To be considered a valid match, the audio-MIDI pair must have a confidence score greater than 0.5. To compute the confidence score for an audio-MIDI pair, Raffel computes the Constant-Q Transform (CQT) [4] for the audio file and the audio-rendered MIDI file, using 48 logarithmically-spaced bins from C2 to B5 (12 bins per octave). Then, the dynamic time warping (`DTW`) algorithm is used to find the optimal alignment, from which the confidence score is directly computed [15]. Although this procedure produces good results, it is extremely slow, as `DTW` has quadratic run-time, which makes this approach intractable.

To speed up the matching process, Raffel proposes learning distance preserving low-dimension embedding spaces, which should allow for highly dissimilar matches to be efficiently removed from the search space. Raffel explores two approaches, an attention-based network $(\mathcal{H}_\infty)$ that embeds arbitrary length CQT matrices into a 128-bit hash code [17], and a convolution-based network $(\mathcal{H}_k)$ that maps $k \times 48$ CQT matrices into 32-bit hash codes [16], which can be used to transform a $n \times 48$ CQT matrix into a sequence of $\lfloor \frac{n}{k} \rfloor$ 32-bit hash codes. Using trained embedding networks $\mathcal{H}_\infty$ and $\mathcal{H}_8$, Raffel employs the following procedure to match a single MIDI CQT m against a set of audio CQTs A.

1. Blocking Stage
   (a) Compute $\mathcal{D}_{\texttt{H}}(\mathcal{H}_\infty(a), \mathcal{H}_\infty(m))$ for each $a \in A$, where $\mathcal{D}_{\texttt{H}}$ is the bitwise hamming distance.
   (b) Construct a set A', containing the $t_1 = 100{,}000$ $a \in A$ that are closest to m, using the distances calculated in 1a.
   (c) Compute $\texttt{DTW}(\mathcal{H}_8(a), \mathcal{H}_8(m))$ for each $a \in A'$.
   (d) Construct a set A'' containing the $t_2 = 250$ $a \in A'$ that are closest to m, using the distances calculated in 1c.
2. Matching Stage
   (a) Compute $\texttt{DTW}(a, m)$ for each $a \in A''$ and record any matches with more than .5 confidence.

[1]https://developer.spotify.com/documentation/web-api/

3.3.1 Modifications to the Matching Procedure

According to Raffel's measurements, it takes an average of 108 seconds on a single CPU to match one MIDI file against 1,000,000 Audio files. As a result, without making modifications to Raffel's procedure, it would take roughly 558 days on a 32-core CPU to match our collections of audio and MIDI files. In order to optimize the audio-MIDI matching procedure to our specific context, we make changes to the blocking stage. Notably, since we do not modify the second stage, and use Raffel's code[2] to compute the confidence scores, our matches can be considered to be the same quality as those found in the LMD.

The simplest modification involved implementing a c++ version of the `DTW` code for 32-bit hash sequences, used in the blocking stage, which runs 2 times faster than Raffel's jit-compiled Cython implementation according to our measurements. We also reconsider the use of the attention based embedding network $\mathcal{H}_\infty$ in Steps 1a and 1b. Using Raffel's approach, Step 1a can be computed very quickly, accounting for less than 1% of the total algorithm run-time. However, due to the low reliability of distance measurements in this embedding space, relatively few audio files can be removed from consideration. As a result, Step 1c takes much longer to run, accounting for roughly half of the total run-time. One reason for the limited accuracy of this approach, is that $\mathcal{H}_\infty$ must embed MIDI and audio CQTs into the same 128-bit hash code, despite MIDI files being much longer than the audio files.

To address this issue, we use $\mathcal{D}_{\mathtt{W}}$, defined in Eq. 3.1, to compute the distances in Step 1a. Given an $n \times 48$ MIDI CQT m and an audio CQT a, we build a set of 30-second length sub-sequences $(\mathcal{X}^m)$ from m, as defined in Eq. 1a, where s is the stride. Using $\mathcal{H}_{128}$ we map each 30-second length CQT matrix (i.e. 646×48) x to a hash code by splitting x into contiguous windowed sub-sequences, computing $\mathcal{H}_{128}(\cdot)$ for each sub-sequence, and concatenating the resulting hash codes. Formally, we refer to this process as $\mathcal{H}^\star_k$, which we define in Eq. 1b, where $\oplus$ denotes concatenation. Then, as shown in Eq. 1c, we compute the bitwise hamming distance $(\mathcal{D}_{\mathtt{H}})$ between $\mathcal{H}^\star_{128}(x)$ and $\mathcal{H}^\star_{128}(a)$ for each $x \in \mathcal{X}^m$, considering the minimum distance to be representative of the distance between m and a.

$$\mathcal{X}^m = \{m[si\!:\!si{+}646] \;:\; 0 \;\leq\; i \;<\; \left\lfloor \frac{n-646+1}{s} \right\rfloor\} \tag{3.1a}$$

$$\mathcal{H}^\star_k(x) = \oplus\{\mathcal{H}_k(x[ki\!:\!k(i{+}1)]) : 0 {\leq} i {<} \left\lfloor \frac{||x||}{k} \right\rfloor\} \tag{3.1b}$$

$$\mathcal{D}_{\mathtt{W}}(a,m) = \min(\{\mathcal{D}_{\mathtt{H}}(\mathcal{H}^\star_{128}(a), \mathcal{H}^\star_{128}(x)) : x {\in} \mathcal{X}^m\}) \tag{3.1c}$$

[2]https://github.com/craffel/midi-dataset

3.3.2 Training the Embedding Networks

We derive our neural network architecture from the one used by Raffel [15]. The first section of the network is comprised of k groups, with each group is containing 2 3×3 convolutional layers, followed by a 2×1 max pooling layer. The second section contains two dense layers with 2048 units each, followed by a 32-dimensional output. The ReLU activation is used in all layers, except for the last layer, which uses the `tanh` activation function to effectively binarize the output. For the $\mathcal{H}_{128}$ network, which learns to downsample a sequence of 128×48 CQT matrix into a 32-bit hash code, there are $k = 5$ groups, using the filter sizes 64, 64, 64, 32, and 16 for each group respectively. For the $\mathcal{H}_8$ network, which learns to downsample a 8×48 CQT matrix into a 32-bit hash code, there are $k = 3$ groups, using 64, 32, and 16 filters per group respectively. We train $\mathcal{H}_{128}$ and $\mathcal{H}_8$ using the same triplet loss as Raffel. In terms of training data, we use the 116,189 audio-MIDI matches from the LMD, which we split into testing, validation and training datasets. We train each network with a learning rate of $1\text{e}{-4}$, and early stopping on validation every 1000 batches, using Keras [6].

3.3.3 Evaluating the Embedding Networks

To evaluate the expected accuracy of distance calculations using our trained embedding networks, we use the same method proposed by Raffel. For a known audio-MIDI pair (m, a), we measure the distance between m and a set of 1,000,000 audio files $\mathcal{X}$, with $a \in \mathcal{X}$, to determine the rank of the correct match. After repeating this process for 1,000 audio-MIDI pairs in our test set, we can measure the proportion of MIDI files where the correct match ranks below a particular threshold. The results are presented in Figure 3.1, including results previously presented by Raffel for purposes of comparison [15]. Although Raffel used different data to train and evaluate the embedding, we can be fairly confident in the reliability of our comparison, as the curve for our $\mathcal{H}_8$ embedding network (Step 1c (Ours)) is nearly identical to the curve for Raffel's $\mathcal{H}_8$ embedding network (Step 1c (Raffel)). Although using $\mathcal{D}_{\texttt{W}}$ slows down Step 1, the results demonstrate that it is much more accurate, which means we can reduce the number of comparisons needed in Steps 1c and 1d, which ultimately speeds up the algorithm, as Step 1c accounts for roughly half of the total run-time.

3.3.4 Matching Against 32,000,000 Audio Files

Clearly, a large factor contributing to the run-time of the matching algorithm is the threshold levels (t_1, t_2) for each stage of the search. Raffel et al. determine t_1 and t_2 based on the evaluation method presented in Figure 3.1. However, this approach is merely a proxy for what we are actually trying accomplish. Put simply, in matching a large collection of MIDI files with a large collection of Audio Files, we are trying to maximize the number of matches. In order to get a sense of the relationship between run-time and the number of matches, we run our matching procedure with 1,000 MIDI files and 10,000,000 audio files, using various thresholds. The results in figure 3.2 show that we pay a high computational cost to increase the number of MIDIs matched. For example, increasing the thresholds from $t_1 =$ 100,000 / $t_2 =$ 250 to $t_1 =$ 1,000,000 / $t_2 =$ 2,500 increases the run-time by

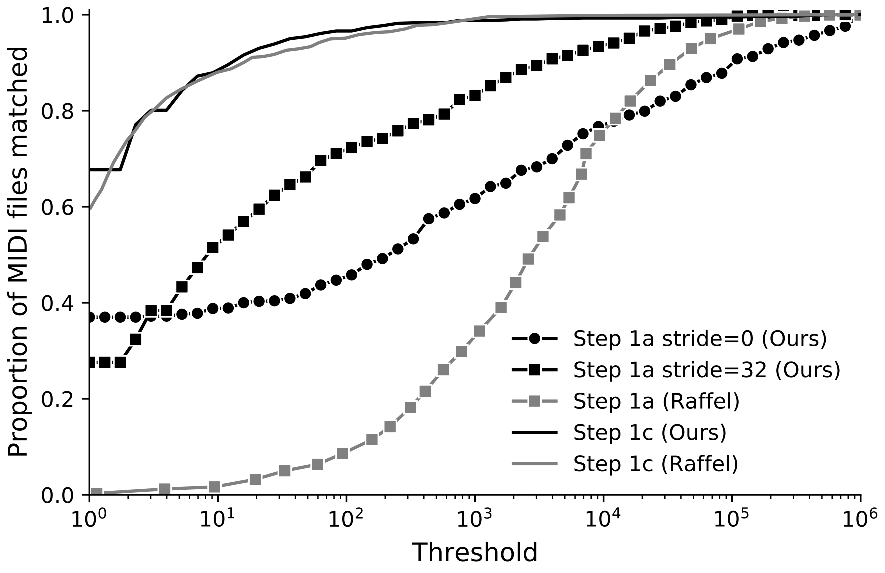


Figure 3.1: Percentage of MIDI files matched at thresholds.

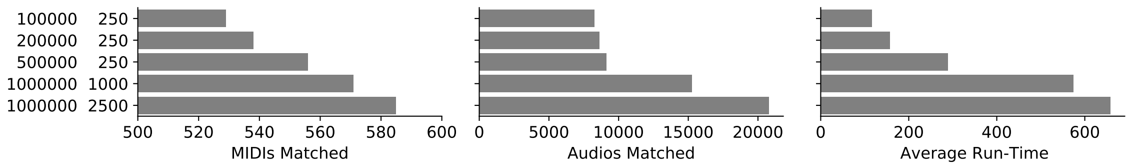


Figure 3.2: The number of MIDI files matched, Audio recordings matched and average match run-time for different thresholds. On the left, the first value denotes t_1 and the second value denotes t_2.

560%, while only yielding a 10% increase in the number of MIDIs matched and a 200% increase in the number of Audio files matched.

Due to memory limitations, it is not possible to match a MIDI CQT against all 32,000,000 audio CQTs at once. As a result, we subdivide the audio CQTs into four chunks, and process them each separately. In light of the results in the previous section, we decided to set $t_1 =$ 100,000 and $t_2 =$ 250 for each chunk. In Table 3.1, we report the results of the Audio-MIDI matching procedure. In comparison to the LMD, where only 26% of the MIDI files were matched to at least one Audio file, we were able to match 56% of the MIDI files, for a total of 237,236 MIDI files matched. Notably, our modifications to the matching procedure also had a substantial impact on the run-time, as the average run-time per match was only 3.3 times more than the run-time for LMD matching, despite matching against over 32 times more audio.

Dataset	MIDIs	Audio Source	Matching Method	Matched MIDIs	Matched Audios	Total Matches	MIDI Match Percentage
LMD	176,581	MSD[1]	Audio	45,129	31,034	116,189	25.6%
MMD	436,631	Spotify	Audio	237,236	2,209,941	10,796,557	52.7%
MMD	436,631	Spotify	Audio+Text	53,496	347,703	829,728	12.3%
MMD	436,631	MusicBrainz	Audio	168,032	1,094,901	8,384,256	38.5%
MMD	436,631	MusicBrainz	Audio+Text	34,174	408,922	1,232,909	7.8%

Table 3.1: Statistics for the audio-MIDI matching. Note that the MusicBrainz matches were computed by combining the Spotify audio-MIDI matches and the Spotify-MusicBrainz links (Section 4). The Percentage of MIDIs Matched column reports the percentage of MIDI files in the respective dataset that have at least one match to an audio file. Total Matches denotes the total number of unique audio-MIDI pairs matched.

3.3.5 High Reliability Audio-MIDI Matches

Although the audio-MIDI matches are fairly reliable, Raffel notes that it is not uncommon for there to be false positives when an audio-MIDI pair share the same chord progression [15]. To address these issues, we produce a subset of the audio-MIDI matches which are more reliable, using artist+title metadata that was collected during the scraping process. In short, we only retain audio-MIDI matches where the title or artist scraped with the MIDI file is a fuzzy match to the metadata on Spotify. Since artists and title metadata frequently contain extraneous information, we remove all content in parenthesis or square brackets, and remove all content following a dash. As a result, the Spotify track titled "Rain Is Falling (Karaoke Version) - Originally Performed By Electric Light Orchestra" would be reduced to "Rain is Falling" after pre-processing. We measure the similarity between two strings using cosine similarity on their tri-gram profiles, and only keep matches when the similarity exceeds .8 for either the artist or the title metadata. Once this procedure has been completed, we are left with 53,496 (12%) matched MIDI files and 829,728 total matches.

3.4 Linking Spotify and MusicBrainz

To further expand the dataset, we make links between Spotify track ids to MusicBrainz recording ids using a classifier trained on audio features. Although AcousticBrainz Labs has provided an archive[3] of the Echo Nest mappings between MusicBrainz and Spotify, we were only able to match 24,363 MIDI files to MusicBrainz IDs using this resource. To train our classifier, we gathered a set of ground truth data using International Standard Recording Codes (ISRC), which are provided by both Spotify and MusicBrainz. Although Spotify provides this information for almost all of their tracks via their API, only a percentage of recordings in the MusicBrainz database have been labeled with an ISRC code. Using the ISRC codes which were available, we were able to compile about 100,000 unique ground truth matches. This data was divided into training, validation and testing sets.

We use the AcousticBrainz API[4] to obtain features for recordings in the MusicBrainz database, since the actual audio is not provided by MusicBrainz or AcousticBrainz. To extract features from the 30-second Spotify preview clips, we use the same feature extractor as AcousticBrainz (Essentia [2]). Using the low-level features extrated via Essentia, we obtain a feature vector of dimension 1773 to represent each audio clip. Then we trained a classifier to predict whether a pair of vectors, one collected from the AcousticBrainz database, and another from Spotify, correspond to the same recording. To train the classifier, we expose the model to ground truth matches, where the AcousticBrainz recording and Spotify recording share the same ISRC, and negative matches, where both recordings do not share the same ISRC. To construct a negative match, we randomly select one recording from each data source (AcousticBrainz and Spotify). Note that for training, validation and testing we make sure the model is exposed to both conditions (ground truth and negative match) an equal number of times.

We use the XGBoost library [5] to train a gradient boosting model. To determine the optimal hyper-parameters for the model, we perform a grid search using the following parameters: nestimators $\{2500, 5000\}$, learning rate $\{.1, .25, .5, .75\}$, and max depth of $\{2, 3, 4\}$. To evaluate the models, we calculate the accuracy with which the model was able to predict if the pair of recordings was a positive (ground-truth) or negative match. We found the model with nestimators=2500 learning rate=.25 and max depth=4 to perform the best on the validation set, achieving 97.6% accuracy. To give us some indication that we are not simply over-fitting on the validation set, we compute the accuracy of the best model using the testing set. Based on the fact that the best model scored 97.5% accuracy on the test set, which was only used once, we can be fairly confident that the model will generalize with this level of accuracy.

Since, at the time of writing, there are 5,534,103 unique recordings in the AcousticBrainz dataset, and 2,209,941 Spotify audio previews (see Table 3.1) which we want to match against,

[3]https://labs.acousticbrainz.org/million-song-dataset-echonest-archive/

[4]https://acousticbrainz.org/data

	Matched Spotify IDs	Matched MusicBrainz IDs	Spotify-MusicBrainz Matches	MIDI-MusicBrainz Matches
MSD Echo Nest	1,307,152	675,240	3,168,164	24,363
ISRC Matches	104,404	69,006	104,404	82,951
Ours	600,142	1,094,901	8,263,482	168,032

Table 3.2: Statistics for the Spotify-MusicBrainz matching.

collecting the model's predictions for each pairwise match would be extremely computationally expensive. To make this process feasible, we first match all the artists in the MusicBrainz database against a list of artists from Spotify using tri-gram cosine distance with a threshold of .7. Then we match each the titles of each recording if the artists were a match, once again using tri-gram cosine distance with a threshold of .7. Then for each potential match, we use the classifier to predict whether it is actually an audio match. Consequently, the error rate should be lower than 2.5% since matches must also have similar metadata (artist title) to be considered a match. The entire process took about 3 days on a single computer. In Table 3.2 below, we outline the results of the Spotify-MusicBrainz linking process. We provide details on the MIDI-MusicBrainz matches which were derived from the audio-MIDI matches in Table 3.1.

3.5 Analyzing the Dataset

3.5.1 Overview Statistics for the Midi Files

In order get a sense of the type of data that was collected, we compute the distributions for several features. We parse a MIDI file into a set of tracks, where a track is simply the set of note onsets and offsets belonging to a (`MIDI track`, `channel`, `instrument`) tuple. Each track is subdivided into a sequence of bars, using the time signature information present in the MIDI file. Due to space limitations, we present a few of the most pertinent features below, providing a more comprehensive overview elsewhere[5]:

1. **Number of Tracks** : The number of tracks, as defined above, in a MIDI file.
2. **Beat Length** : The total length in quarter note beats of an entire MIDI file.
3. **Notes Per Bar** : The number of note onsets occurring in a bar. We measure this on each track separately, so that we do not conflate notes per bar and number of tracks.

We compute the distribution of each of these features across three different sets of data: the LMD, MMD, and their symmetric difference MMD Δ LMD. These distributions are shown in Figure 3.3. On a whole, the graphs demonstrate that LMD, MMD Δ LMD and MMD are all fairly similar, however there are some differences worth noting. The two most obvious differences, are

[5]https://github.com/jeffreyjohnens/MetaMIDIDataset

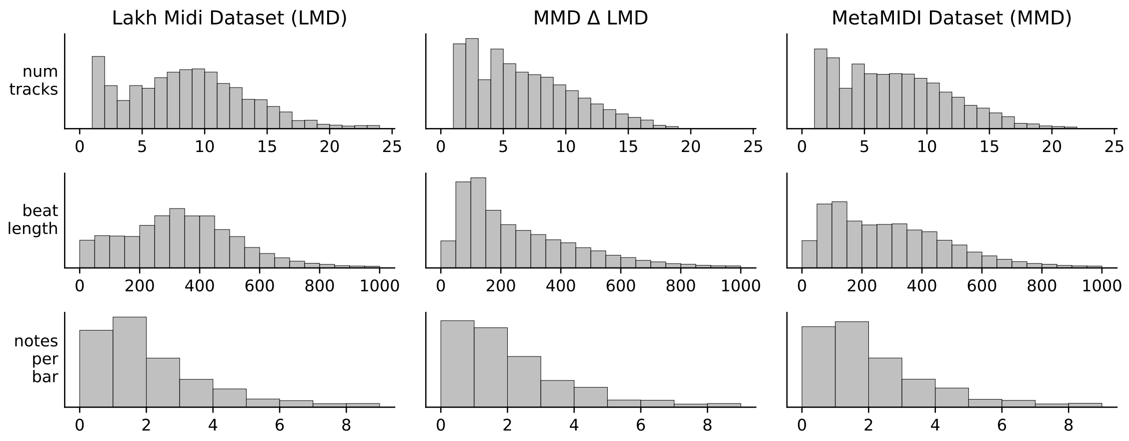


Figure 3.3: The distributions for various features computed on LMD, MMD Δ LMD and MMD.

the beat length and number of tracks. The difference in beat length distributions can mainly be explained by the fact that two of the sites we scraped MIDIs from only provide 30s preview MIDI clips for free. Since the musical quality of these shorter MIDIs is comparable to that found in the LMD, we saw no real reason to exclude these files. The difference in track counts per MIDI does not have an obvious explanation, but is worth noting nonetheless.

3.5.2 Estimating the Reliability of Scraped Metadata

To gauge the reliability of the scraped metadata, we analyze instances where metadata was collected for the same MIDI file (md5 checksum) from multiple sources. In total, there are over 10,000 MIDI files which satisfy this criteria. For each of these MIDI files, we compare the title/artist and genre/category metadata separately. For the title/artist metadata, we concatenate this metadata into a single string, delimited by a "-", and compute cosine similarity on their tri-gram profiles. For the genre metadata, we compute tri-gram cosine similarity between each pairwise combination of elements between two genre/category lists, and report the maximum similarity. The mean similarity is 73.7% for title/artist metadata and 1.1% for genre metadata. Immediately apparent, is the significant discrepancy, as title/artist metadata appears to be fairly consistent from site to site, while genre metadata is not. Further manual analysis reveals that the genres/categories are often very generic, which may make them unsuitable for some purposes. In some respects, this is not altogether surprising, as determining the genre/category of a piece of music is a highly subjective process, and other research has shown a significant level of disagreement [3]. However, with regards to the artist/title metadata, these results seem to indicate that we can be fairly confident in this form of metadata. It is worth noting that this type of analysis does not rule out cases where artist/title metadata on multiple sites was derived from a single inaccurate source to begin with.

3.5.3 False Positives and Audio Midi Matching

Using the standard and high-reliability sets of audio-MIDI matches, we can further analyze the source of false positives in the matching procedure. To do this, we compare the genre distribution of each set of audio-MIDI matches. Since Spotify uses more than 5,000 genres, many of which contain descriptors of particular locations (ex. Louisville Indie) or languages (ex. Spanish Indie Pop), we pre-process the data to remove geographical locations, demonyms and languoids. This results in about 2,500 genres. To further aggregate these genres into broader categories we employ a graph embedding approach. Using the Spotify API, we collect a list of genres for 336,507 different artists. For example, the band U2 has a genre list containing three genres: Irish Rock, Permanent Wave, and Rock. Note that after we apply our pre-processing procedure, U2 has two genres: Rock and Permanent Wave. Of particular interest for our purposes here, is artists which have a genre list containing more than one genre, as the overall frequency with which two genres co-occur within genre lists should provide a good indication of their similarity.

Then we construct a graph where each genre is a node, and the edge weights between nodes are the count of co-occurrences within the genre lists. To create the embedding, we use the Node2Vec algorithm [11], which creates an embedding space that is trained on relations found within the graph. Similar to the word2vec algorithm [14], where adjacent groupings of words inform the embedding, random walks on the graph are used to infer a context for each node. We use the nodevectors[6] implementation of Node2Vec to learn a 32-dimensional embedding space, training with random walks of length 30 for 100 epochs. To determine a small set of k representative genres, we use Agglomerative Hierarchical Clustering with Ward linkage to partition the embedded genre vectors into k clusters. In order to give each cluster a human-readable label, we count the frequency with which each of the genres belonging to the cluster is used in the genre lists. The most frequently used genre is taken as the label for each genre. We set $k = 15$, which produces the following set of genres: indie, rock, experimental, jazz, pop, metal, musica, electronic, folk, choir, classical, punk, punk rock, hip-hop, and electronica. We admit that our decision to set $k = 15$ is fairly arbitrary, however, due to the nature of our clustering procedure, selecting a different value for k would not have a large impact. For example, setting $k = 16$ produces the same set of 15 genres with one new genre cluster.

In Figure 3.4 the genre distributions are plotted for each version of the matching procedure. Since we can be fairly confident that the audio + text matches are more accurate, analyzing the discrepancies between the genre distributions can help identify some of the shortcomings of the `DTW` audio-MIDI match algorithm. In the audio matches distribution, we see a large increase in pieces classified as pop and electronic, which indicates these pieces are likely the source of most of the error. This may be a byproduct of their simple harmonic structure, and/or the prevalence of remixes and covers within these particular genres.

[6]https://github.com/VHRanger/nodevectors

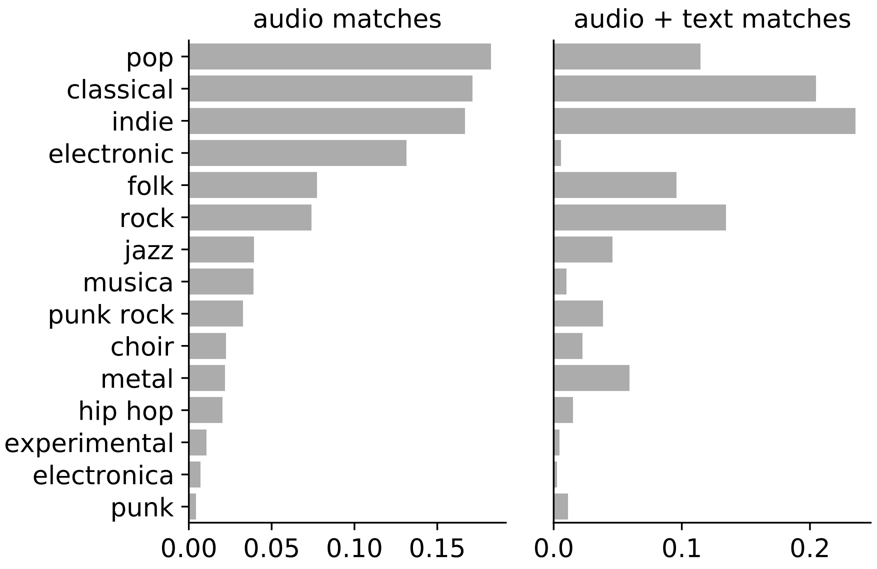


Figure 3.4: The distribution of genres for matched MIDI files using two methods: audio and audio + text.

3.6 Using the MetaMIDI Dataset

The dataset, as well as a detailed description of its contents, can be accessed through the MetaMIDI Dataset repository[7]. Throughout the dataset, MIDI files are identified by their md5 checksum. We provide mappings from md5 checksums to Spotify track ids and MusicBrainz recording ids, which can be used to access a plethora of metadata. The MusicBrainz database provides access to variety of linked entities including artists, recordings, releases, composers, producers, recording engineers and labels. Detailed attributes are available for most entities. For example, the MusicBrainz entry for the group Bon Iver, provides the date and location where the group was established, a list of aliases, a set of genre tags, and a comprehensive list of links to external websites. Using the Spotify API, a variety of metadata can be accessed, including track-based audio features such as danceability, valence, liveness and energy; and additional metadata ranging from genre to artist popularity.

3.7 Conclusion

Although the primary contribution is the dataset itself, we have also provided reusable insights related to the audio-MIDI matching algorithm and the Spotify-MusicBrainz linking procedure. One limitation worth noting, is the uncertainty in relying on Spotify's 30-second clips to persist into the future, which unfortunately has already become an issue with the 7Digital clips in the Million Song Dataset [1]. With regards to the dataset, we anticipate a wide variety of potential use-cases for this

[7]https://github.com/jeffreyjohnens/MetaMIDIDataset

data. Since many generative systems have been trained using the Lakh MIDI Dataset [7, 8, 18], the MMD will undoubtedly be a valuable asset to research in this area, as it features 2.4 times more MIDI files. More broadly, the metadata that our audio-MIDI matches provide access to, as well as the audio-MIDI matches themselves, can be used to support a variety of scientific inquires related to MIR and Musicology.

Bibliography

[1] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. "The Million Song Dataset". In: *Proc. of the 12th International Society for Music Information Retrieval Conference*. 2011.

[2] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez Gutiérrez, Sankalp Gulati, Herrera Boyer, Oscar Mayor, Gerard Roma Trepat, Justin Salamon, José Ricardo Zapata González, Xavier Serra, et al. "Essentia: An audio analysis library for music information retrieval". In: *Proc. of the 14th International Society for Music Information Retrieval Conference*. 2013.

[3] Romain Brisson and Renzo Bianchi. "On the relevance of music genre-based analysis in research on musical tastes". In: *Psychology of Music* 48.6 (2020), pp. 777–794.

[4] Judith C Brown. "Calculation of a constant Q spectral transform". In: *The Journal of the Acoustical Society of America* 89.1 (1991), pp. 425–434.

[5] Tianqi Chen and Carlos Guestrin. "XGBoost: A scalable tree boosting system". In: *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 785–794.

[6] François Chollet et al. *Keras*. https://keras.io. 2015.

[7] Chris Donahue, Huanru Henry Mao, Yiting Ethan Li, Garrison W Cottrell, and Julian McAuley. "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training". In: *Proc. of the 20th International Society for Music Information Retrieval Conference*. 2019, pp. 685–692.

[8] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018, pp. 34–41.

[9] Andres Ferraro and Kjell Lemström. "On large-scale genre classification in symbolically encoded music by automatic identification of repeating patterns". In: *Proc. of the 5th International Conference on Digital Libraries for Musicology*. 2018, pp. 34–37.

[10] Francesco Foscarin, Andrew Mcleod, Philippe Rigaux, Florent Jacquemard, and Masahiko Sakai. "ASAP: a dataset of aligned scores and performances for piano transcription". In: *Proc. of the 21st International Society for Music Information Retrieval Conference*. 2020.

[11] Aditya Grover and Jure Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 855–864.

[12] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse H. Engel, and Douglas Eck. “Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset”. In: *7th International Conference on Learning Representations*. 2019.

[13] Ethan Manilow, Gordon Wichern, Prem Seetharaman, and Jonathan Le Roux. “Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity”. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 2019, pp. 45–49.

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[15] Colin Raffel. “Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching”. PhD thesis. Columbia University, 2016.

[16] Colin Raffel and Daniel PW Ellis. “Large-Scale Content-Based Matching of MIDI and Audio Files.” In: *Proc. of the 16th International Society for Music Information Retrieval Conference*. 2015, pp. 234–240.

[17] Colin Raffel and Daniel PW Ellis. “Pruning subsequence search with attention-based embedding”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2016, pp. 554–558.

[18] Adam Roberts, Jesse H. Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. “A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music”. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 4361–4370.

[19] Hendrik Schreiber and Meinard Müller. “A Single-Step Approach to Musical Tempo Estimation Using a Convolutional Neural Network.” In: *Proc. of the 19th International Society for Music Information Retrieval Conference*. 2018, pp. 98–105.

Chapter 4

CAEMSI : A Cross-Domain Analytic Evaluation Methodology for Style Imitation

As published in Ens, J. & Pasquier, P. (2018). *CAEMSI : A Cross-Domain Analytic Evaluation Methodology for Style Imitation.* International Conference on Computational Creativity (ICCC). (pp. 64-71).

Abstract

We propose CAEMSI, a cross-domain analytic evaluation methodology for Style Imitation (SI) systems, based on a set of statistical significance tests that allow hypotheses comparing two corpora to be tested. Typically, SI systems are evaluated using human participants, however, this type of approach has several weaknesses. For humans to provide reliable assessments of an SI system, they must possess a sufficient degree of domain knowledge, which can place significant limitations on the pool of participants. Furthermore, both human bias against computer-generated artifacts, and the variability of participants' assessments call the reliability of the results into question. Most importantly, the use of human participants places limitations on the number of generated artifacts and SI systems which can be feasibly evaluated. Directly motivated by these shortcomings, CAEMSI provides a robust and scalable approach to the evaluation problem. Normalized Compression Distance, a domain-independent distance metric, is used to measure the distance between individual artifacts within a corpus. The difference between corpora is measured using test statistics derived from these inter-artifact distances, and permutation testing is used to determine the significance of the difference. We provide empirical evidence validating the statistical significance tests, using datasets from two distinct domains.

Keywords: Generative Music; Evaluation; Machine Learning; Big Data

4.1 Introduction

There is growing demand for creative generative systems in the entertainment industry, which has prompted an abundance of research in the area of Style Imitation (SI). Given a corpus $C = \{c_1, ..., c_n\}$, SI systems aim to generate new artifacts that emulate the stylistic characteristics of C. Many of these SI systems generate some form of musical content, including; harmonic progressions, melodies [42], and polyphonic compositions [22]. A more comprehensive overview of the work in the domain can be found elsewhere [29, 5]. In the visual art domain, the Creative Adversarial Network (CAN) is trained to generate visual art that deviates from the styles it has already learned [13]. Moreover, many Natural Language Generation (NLG) systems have been developed that generate jokes, poetry, and narratives in a particular style [16]. To accommodate the large influx of generative systems in recent years, we propose CAEMSI [1].

Ritchie mentions two conditions for determining if creativity has occurred: *novelty*, the degree to which an artifact is dissimilar to other examples within the corpus and *quality* [34]. He also emphasizes the notion of *typicality*, the degree to which a generated artifact is representative of the source corpus (C). In the context of style imitation, measuring typicality is of critical importance, as the performance of an SI system hinges on its ability to emulate the stylistic characteristics of the source corpus. As a result, CAEMSI focuses on measuring the typicality of a generated corpus, with respect to the source corpus. Although novelty is also an important indicator of the system's quality, as it is generally undesireable for an SI system to plagiarize large sections from the source corpus, we leave this aspect of evaluation for future work.

Traditionally, participants assess the capacity of a particular system to emulate a particular style, allowing researchers to make claims about the success of that system. Unfortunately, this is not a scalable solution, and can make it difficult to compare SI systems. With the long-term goal of creating highly capable SI systems, it is necessary to develop robust methods for the evaluation of these systems, as a lack of methodical evaluation can have a negative effect on research progress [31]. The approach described in this paper is domain independent, harnessing the power of Normalized Compression Distance (NCD) [8] and permutation testing to provide a scalable solution to the problem of SI system evaluation. In order to demonstrate the effectiveness of this approach, we conduct experiments on datasets in two different domains; the Wikiart image dataset[2] and the Classical Archives MIDI dataset[3].

[1]The code is available https://goo.gl/ejN1RM

[2]https://www.wikiart.org/

[3]https://www.classicalarchives.com/midi.html

4.2 Evaluation Methodologies

Although many methodologies that evaluate the creative capacity of a generative system have been proposed, we will limit our discussion to those which have been used to measure typicality. In general, we can divide these methodologies into two categories, those which rely on human participants, and those based purely on computation. Unto our knowledge, the only statistical evaluation methodology for typicality was proposed by Thomas et al., however it is only capable of evaluating melodic composition systems [39].

The Consensual Assessment Technique (CAT) [1] is based on the notion that experts are the most capable of distinguishing creative artifacts within their respective domain. To account for discrepancies, which arise given the subjective nature of these assessments, the CAT averages the assessments of several experts. Pearce et al. employ the CAT to evaluate the success of melodic generation algorithms [32].

Another approach, inspired by the Turing-test, measures participants ability to discriminate between computer-generated artifacts and artifacts from the source corpus. This evaluation methodology has been used to evaluate many SI systems, including a Deep LSTM Network that generates Bach chorales [22], and a Generative Adversarial Network that generates images [13].

4.3 Related Work

With regards to typicality, Conklin [10] provides a statistical framework to measure distinctiveness, which can be understood as the inverse of typicality. Although the pattern representation that Conklin employs is designed for musical data, this approach could be adapted to other domains provided a suitable pattern based representation is specified. More generally, several quantitative metrics for creative systems have been proposed. Maher has proposed two metrics for measuring creativity quantitatively. The first, equates *novelty* with distance from predominant clusters of artifacts, measures *surprise* using pattern matching algorithms, and calculates *value* using a fitness function [23]. However, it is not clear how the proposed metrics would be applied to an arbitrary domain, and no proof of concept is provided. The second, uses Bayesian inference to measure the novelty of an artifact, which is used to evaluate potential designs for laptop computers [24].

Burns measures creativity as the combination of psychological arousal, which is computed using Shannon entropy, and appraisal, which is computed using Bayesian theory [6]. The Regent-Dependent Creativity (RDC) metric measures value and novelty. Artifacts are represented by a set of pairs $(P(\textit{regent}, \textit{dependent}))$, where *regent* is an action or attribute, and *dependent* is a state or target for an action [14]. Using a graph, which includes associations between artifacts, they propose metrics to measure synergy, the value produced by various elements acting cooperatively, and Bayesian surprise, the degree to which an artifact is unexpected or novel. Although this metric seems to work well for the low dimensional problems presented in the paper, it is not clear that this approach could efficiently handle artifacts which require a large number of pairs for representation.

Furthermore, it relies on the domain knowledge of synergy, which is difficult to determine in some domains.

4.4 Motivation

Although human-based evaluation methodologies are not without their strengths, the shortcomings of these methodologies directly motivated the development of the statistical tests proposed in this paper.

4.4.1 Domain Knowledge

Accurately assessing the typicality of an artifact with respect to a source corpus, requires a significant amount of domain knowledge, as the participant must be familiar with the stylistic characteristics of the source corpus. This issue is exacerbated when performing a CAT, since participants must have an expert level knowledge of the source corpus. Undoubtedly, this is one of the primary reasons an abundance of musical SI systems have focused on imitating Bach chorales, as there is a large pool of experts, and most people are familiar with Bach's work. Since a lack of domain knowledge undermines the reliability of the evaluation process, the types of scientific inquiries which have been explored are biased by restrictions on the source corpora, placing limitations on scientific progress in this area.

4.4.2 Bias Against Generative Systems

Previous research has shown that when participants were asked to distinguish between two folk melodies, some of which were human-composed and others which were recombinations of the human-composed melodies, participants attributed unusual or disagreeable human compositions to the computer [11]. Norton found a significant bias against images labeled as being generated by a computer [27]. In contrast, several studies have demonstrated that the knowledge that a computer created a piece of music, does not significantly affect the participants' evaluation and enjoyment of the piece [26, 15, 28]. Although Moffat's study did not explicity test the same hypothesis as Dahlig, their results corroborate the same conclusion, as participants attributed compositions they disliked to the computer, independent of their actual authorship.

When participants are tasked with making the distinction between human-generated and computer-generated artworks, they may in fact be searching for features which they expect to be generated by a computer, rather than focusing on the broader style of the composition [2]. As a result, the test degenerates to one which is focused on counting perceived mistakes. This issue has been highlighted by Pearce in his discussion on the evaluation of musical composition systems [30]. Clearly, this type of bias is very problematic when attempting to evaluate an SI system that imitates artifacts that humans tend to find disagreeable, such as the atonal works of Arnold Schoenberg.

4.4.3 Variability

The subjective nature of creativity-based assessments poses problems for the systematic evaluation of creative systems in general. There is evidence that cultural background can have an effect on how an artifact is perceived. For example, Eerola found that western and African listeners perceived musical attributes differently [12]. Furthermore, environmental factors will affect the reliability of these assessments, including the equipment used to observe the artifact, and the physical condition of the participant. Although those who design experiments take many steps to mitigate the effects of these factors, Schedl at al. [36] provide evidence that inter-rater agreement is still limited in a practical setting. In one case, non-experts' assessments of poetry were found to be negatively correlated with the assessments of experts [19]. Similarly, Kaufman found that experts were far more reliable than non-experts, when asked to judge the creativity of a short story, as measured by inter-rater reliability for both groups [17].

4.4.4 Scalability

Unfortunately, using human participants places limitations on the total number of assessments that can be collected. Participants are only capable of making so many assessments before fatigue will begin to degrade the quality of their responses. Notably, this problem is exacerbated by the limited number of participants involved when conducting a *CAT*. Although crowdsourcing does make it easier to collect a large number of assessments, there are still monetary and time limitations that place restrictions on the the total number of assessments that can be feasibly collected. Clearly, the limited scalability of these evaluation methods is in direct conflict with the large number of artifacts which generative systems can produce.

In many cases, a small subset of the generated artifacts is used to evaluate the system, decreasing the number of assessments required. However, issues will naturally arise when the selected subset is not adequately representative of the system's output as a whole [2]. Moreover, it is not trivial to determine if a subset of artifacts is representative of the systems output a priori. Most importantly, these limitations make it increasingly difficult to evaluate a large number of systems.

4.4.5 The Proposed Solution

In contrast to human-based evaluation methods, CAEMSI eschews the issues of domain knowledge, human bias, and variability. Admittedly, there are still limitations with respect to the size of corpora, which will be addressed in future work. However, computation based methods of evaluation are far more scalable than human-based solutions, as computers can process artifacts much faster than humans can.

4.5 Statistical Tests for Typicality

In what follows, $X = [x_i, i = 1, ..., n]$ denotes a vector X, containing n elements. $X \oplus Y$ denotes the concatenation of two vectors. We use the term *corpora* to denote a vector of binary strings. $\mu(X)$ denotes the mean of a vector X, while $\phi(X)$ denotes the median. p_{diff} and p_{eqv} denote the significance of the statistical test for difference and equivalence respectively.

Given two corpora, $A = [a_i, i = 1, ..., n]$ and $B = [b_i, i = 1, ..., m]$, we test the null hypothesis $H_{D0} : A = B$ ($p_{\text{diff}} > \alpha$) against $H_{D1} : A \neq B$ ($p_{\text{diff}} \leq \alpha$) and the null hypothesis $H_{E0} : A \neq B$ ($p_{\text{eqv}} > \alpha$) against $H_{E1} : A = B$ ($p_{\text{eqv}} \leq \alpha$). When the result of a statistical test is insignificant, we accept the null hypothesis, which only indicates that there was insufficient evidence to support the alternate hypothesis, and does not validate or invalidate the null hypothesis. As a result, accepting the null hypothesis $H_{D0} : A = B$ is not the same as rejecting the null hypothesis $H_{E0} : A \neq B$ and accepting the alternative hypothesis $H_{E1} : A = B$, as only the latter indicates that $A = B$. Consequently, we can determine if $A = B$ using p_{eqv} and if $A \neq B$ using p_{diff}.

4.5.1 Normalized Compression Distance

Put simply, the *Kolmogorov complexity* $(K(x))$ of a finite length binary string x is the minimum number of bits required to store x without any loss of information. More formally, $K(x)$ denotes the length of the shortest Universal Turing Machine that prints x and stops [38]. Intuitively, the minimum number of bits required to store a random string would be close to the number of bits used to represent the original string. As a result, a random string would have a high Kolmogorov complexity. In contrast, a string with a large number of repeated subsequences, would have a low Kolmogorov complexity. Although Kolmogorov complexity provides an absolute lower bound on the compression of a string, $K(x)$ is non-computable [20], so a real-world compressor is used to approximate $K(x)$ in practice.

The *conditional Kolmogorov complexity* $(K(x|y))$ of a string x relative to a string y, denotes the length of the shortest program that prints x and stops, with y provided as additional input to the computational process. For example, if $x \simeq y$, $K(x|y)$ would be very small, as the program could reproduce x from y without requiring much additional information. In contrast, if x and y are highly dissimilar, $K(x|y)$ would be quite large.

Information distance is the length of the shortest binary program that can compute x from y and y from x. As a result, when x and y have a lot of mutual information, the length of this program will be fairly short. Li et al. propose the *normalized information distance* (4.1).

$$d(x, y) = \frac{\max(K(x|y), K(y|x))}{\max(K(x), K(y))} \tag{4.1}$$

Since $K(x|y) \simeq K(xy) - K(x)$ [20], where xy denotes the concatenation of strings x and y, we can reformulate (4.1) to arrive at a computable *normalized compression distance* (NCD) (4.2). In practice, $K(x)$ is the length of string produced by a real-world compression algorithm, such as zlib. Although we tested several compression algorithms, we did not notice significant variation in terms of performance.

$$D(x, y) = \frac{K(xy) - \min(K(x), K(y))}{\max(K(x), K(y))} \tag{4.2}$$

Li et al. demonstrate that *NCD* is a universal distance metric, satisfying the following constraints.

1. $D(x, y) = 0 \, \textit{iff} \, x = y$ (Identity)

2. $D(x, y) + D(y, z) \geq D(x, z)$ (Triangle Equality)

3. $D(x, y) = D(y, x)$ (Symmetry)

Notably, *NCD* has been applied to problems in a variety of domains, including music classification [7, 21], protein sequence classification [18], image registration [4], and document classification [3]. Others used *NCD* to evaluate *machine translation* (MT) by measuring the distance between the predicted translation and the ground truth translation [40].

4.5.2 Distance Matrix Construction

Given a valid distance metric D and two corpora ($A = [a_i, i = 1, ..., n]$ and $B = [b_i, i = 1, ..., m]$), we can construct a pairwise distance matrix M, where $M_{ij} = D(c_i, c_j)$, and $C = A \oplus B = [c_i, i = 1, ..., n+m]$. We use several subsets of M to perform the proposed statistical tests. In the formula below, w_A and w_B are vectors containing all distinct within group distances for corpora A, and B respectively, while $b_{A,B}$ contains all between group distances. Notably, $l = n + m$ in the equations below.

$$w_A = [M_{ij}, i = 1, ..., n; j = 1, ..., n; j > i] \tag{4.3}$$

$$w_B = [M_{ij}, i = n+1, ..., l; j = n+1, ..., l; j > i] \tag{4.4}$$

$$b_{A,B} = [M_{ij}, i = 1, ..., n; j = n+1, ..., l] \tag{4.5}$$

4.5.3 Permutation Testing

A *permutation test* is a statistical significance test which requires no prior knowledge about the distribution of the test statistic under the null hypothesis, as this distribution is generated by calculating the test statistic for each possible labelling of the data. For example, consider the vector $C = A \oplus B$, which is comprised of two corpora delineated by the labels $\mathbf{L} = [l_i, i = 1, ..., n + m; l_{i \leq n} = 0, l_{i > n} = 1]$, and a test statistic $S = \mu(C_0) - \mu(C_1)$, where $C_j = \{c_i \,|\, l_i = j\}$. First, compute S using $\mathbf{L}$. Then compute S for each possible permutation of $\mathbf{L}$ to construct the distribution under the null hypothesis. Since the number of permutations grows exponentially, as comparing two corpora of size 50 would require $\binom{100}{50} \simeq 10^{29}$ distinct permutations, we approximate this procedure by randomly selecting m permutations. This procedure accommodates complex test statistics, for which it would be intractable, or overly difficult, to compute the distribution of the test statistic under the null hypothesis.

4.5.4 Testing for Difference

To test the hypothesis that two corpora are different, we adapt a permutation testing framework that was used to compare two groups of brain networks [37]. Simpson et al. create a pairwise distance matrix M using the Kolmogorov-Smirnov statistic, however, we use *NCD* instead.

$$R(M) = \frac{\mu(b_{A,B})}{\mu(w_A \oplus w_B)} \tag{4.6}$$

When R is greater than 1, the average between group distance is greater than the within group distance. Therefore, $R > 1$ suggests that the two corpora are likely distinct. In contrast, when $R \simeq 1$, there is likely no difference between the two corpora. The proposed test is detailed in the steps below, where $\mathbf{I}(\cdot) = 1$ if $(\cdot)$ is true and 0 otherwise.

1. Given two corpora $A = [a_i, i = 1, ..., n]$ and $B = [b_i, i = 1, ..., m]$, create a pairwise distance matrix M using (4.2).
2. Calculate the test statistic $T = R(M)$ using (4.6).
3. Take a random permutation (u^*) of the ordering $u = (1, ..., n+m)$ and reorder the columns and rows using this ordering to create M^*.
4. Calculate the test statistic $T^* = R(M^*)$ using (4.6).
5. Repeat steps 3 and 4 N times, producing the output $[T^*_n, n = 1, ..., N]$.
6. Calculate the p-value, $p_{\text{diff}} = \sum_{n=1}^{N} \mathbf{I}(T^*_n \geq T)/N$.

4.5.5 Testing for Equivalence

The proposed test for the equivalence of two corpora, is based on the following assumption.

$$(w_A = b_{A,B}) \wedge (w_B = b_{A,B}) \implies A = B \tag{4.7}$$

The intuition behind this assumption is shown in Figure 4.1 and 4.2, which show the cumulative distributions of w_A, w_B, and $b_{A,B}$ for an intra-artist comparison and an inter-artist comparison respectively. When two distinct corpora are compared, $b_{A,B} \neq w_A$ and $b_{A,B} \neq w_B$, as shown in Figure 4.1. In contrast, when two similar corpora are compared, $b_{A,B} \simeq w_A \simeq w_B$, as shown in Figure 4.2. In practice, the distributions of w_A, w_B, and $b_{A,B}$ are frequently skewed, and sometimes multi-modal, which necessitates a non-parametric test for equivalence.

As a result, we employ a permutation testing framework [33], which is based on Roy's Union-Intersection approach [35], to test for the equivalence of two distributions. First, it is necessary to define an equivalence interval on which the two distributions will be considered equal. ε_I and ε_S denote the inferior and superior margins, respectively. Then we test two hypotheses; $H_{I0} : \delta \geq -\varepsilon_I$ against $H_{I1} : \delta < -\varepsilon_I$ and $H_{S0} : \delta \leq \varepsilon_S$ against $H_{S1} : \delta > \varepsilon_S$, where δ is the divergence between the two distributions being compared. In some cases, this is measured as the difference between the means (μ), however we use the difference between the medians (ϕ), as it is more robust to outliers. As a result, the global null hypothesis (H_{E0}) is true if both one-sided null hypotheses (H_{I0}, H_{S0}) are true, and the global alternative hypothesis (H_{E1}) is true if at least one of H_{I1} and H_{S1} is true. The following algorithm is used to test for the equivalence of two distributions.

1. Given two vectors $F = [f_i, i = 1, ..., n]$ and $G = [g_i, i = 1, ..., m]$, compute the rank transform of $F \oplus G$ to derive a rank transformed version F and G.

2. Given the superior and inferior equivalence margins $(\varepsilon_I, \varepsilon_S)$, we create two vectors $X_I = F \oplus (G + \varepsilon_I)$ and $X_S = F \oplus (G - \varepsilon_S)$, and an ordering $u = (1, ..., n{+}m)$.

3. Compute the test statistic for both hypothesis $T_I = \phi(X_{IF}) - \phi(X_{IG})$ and $T_S = \phi(X_{SG}) - \phi(X_{SF})$ where

$$\begin{aligned} X_{IF} &= [X_I(u_i), i = 1, ..., n] \\ X_{IG} &= [X_I(u_i), i = n{+}1, ..., n{+}m] \\ X_{SF} &= [X_S(u_i), i = 1, ..., n] \\ X_{SG} &= [X_S(u_i), i = n{+}1, ..., n{+}m] \end{aligned}$$

   and $X(j)$ denotes the j_{th} element in X.

4. Take a random permutation (u^*) of the ordering u.

5. Compute the test statistics using the ordering u^*. $T_I^* = \phi(X_{IF}) - \phi(X_{IG})$ and $T_S^* = \phi(X_{SG}) - \phi(X_{SF})$.

6. Repeat steps 3 and 4 N times to simulate the distribution of the two partial test statistics, producing the output $[(T_{In}^*, T_{Sn}^*), n = 1, ..., N]$.

7. Compute the two partial test statistics $\lambda_h = \sum_{n=1}^{N} \mathbf{I}(T_{hn}^* \geq T_h)/N$ for $h = I, S$. Then the global test statistic is $\lambda(F, G) = \max(1 - \lambda_I, 1 - \lambda_S)$.

To test for the equivalence of two corpora, we compute the distance matrix M using *NCD*, then we compute (4.8). As a result, if both $\lambda(w_A, b_{A,B})$ and $\lambda(w_B, b_{A,B})$ are significant, then we consider the two corpora equivalent.

$$p_{\text{eqv}} = \max(\lambda(w_A, b_{A,B}), \lambda(w_B, b_{A,B})) \tag{4.8}$$

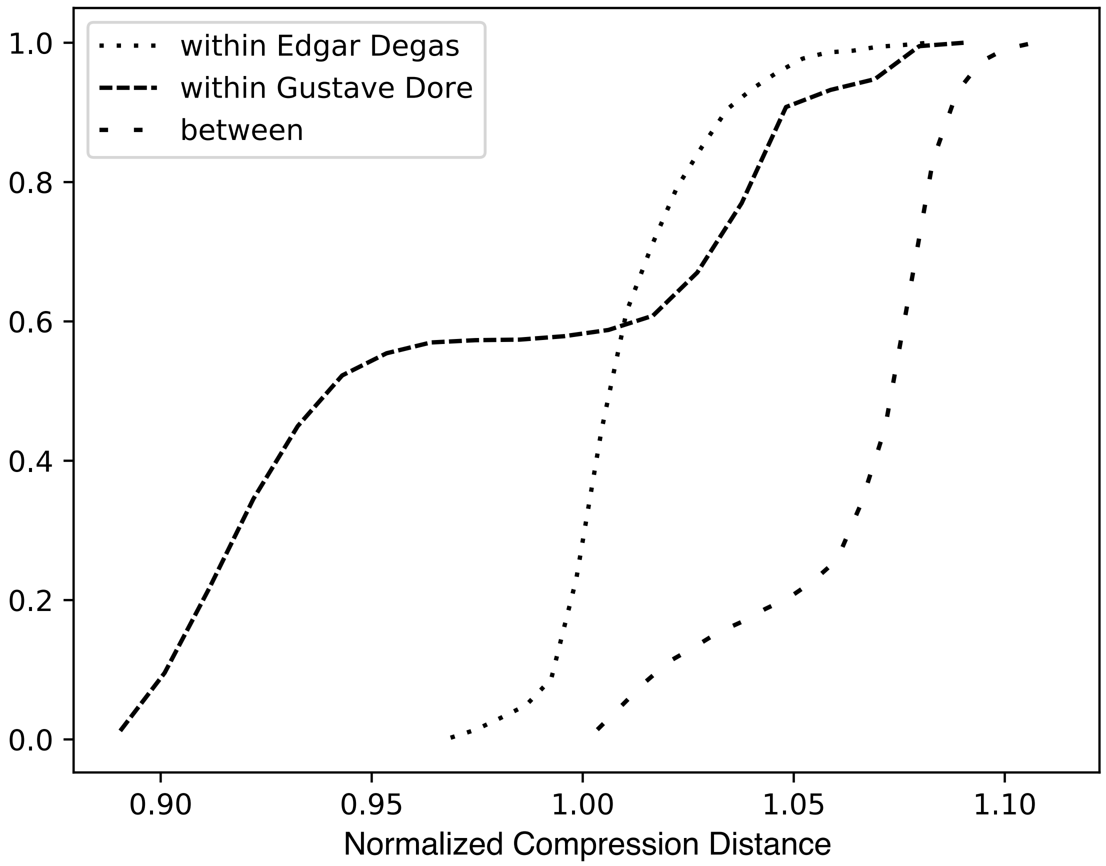


Figure 4.1: The cumulative *NCD* distributions (w_A, w_B, and $b_{A,B}$) used to compare 50 of Edgar Degas' (A) artworks and 50 of Gustave Dor's (B) artworks.

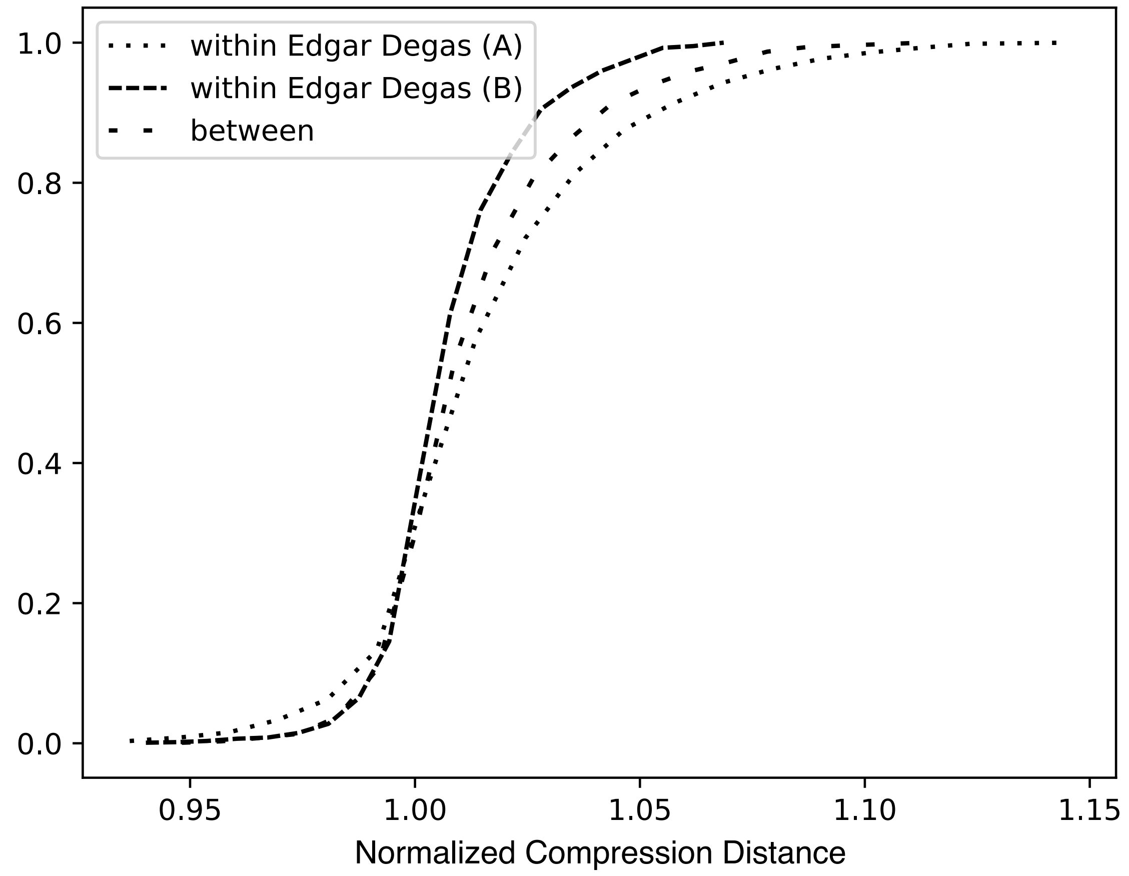


Figure 4.2: The cumulative *NCD* distributions (w_A, w_B, and $b_{A,B}$) used to compare two disjoint subsets of Edgar Degas' artwork, both of size 50.

4.6 Experiment

4.6.1 Methodology

To evaluate the proposed statistical tests, we use datasets from two different domains; the classical archives MIDI dataset, which consists of 14,724 compositions by 843 distinct composers, and the Wikiart dataset, which consists of 19,052 paintings by 23 artists. There are two conditions, one where both corpora (A, B) have the same class (they are created by the same composer or artist), and another where the corpora have a different class. Therefore, the ground truth is calculated using (4.9), and the condition predicted by each statistical test is calculated using (4.10), with the standard significance level $(\alpha = 0.05)$. To create corpora of different sizes, we randomly select artifacts without replacement belonging to the same class.

$$g(a,b) = \begin{cases} 0, & \text{if class}(a) \neq \text{class}(b) \\ 1, & \text{else} \end{cases} \tag{4.9}$$

$$\hat{g}(a,b) = \begin{cases} 0, & \text{if } p_{\text{eqv}} \geq \alpha \text{ or } p_{\text{diff}} < \alpha \\ 1, & \text{if } p_{\text{eqv}} < \alpha \text{ or } p_{\text{diff}} \geq \alpha \end{cases} \tag{4.10}$$

4.6.2 Data Pre-Processing

Since our test statistic takes the pairwise distance of all items within a corpora into consideration, having a number of duplicate items would artificially decrease values in w_A and w_B. As a result, we took the following steps to remove duplicate items in each dataset.

1. Remove all artifacts which belong to the same class and have the same title.
2. Remove all artifacts which belong to the same class and have a similarity greater than a threshold (t_{sim}).

We measure the similarity between two images using the structural similarity index [41], which takes structural information into account, rather than quantifying visible differences. To measure the similarity of two MIDI files, we extract a list of the pitches in the MIDI file ordered by onset time. Given compute time constraints, we only take the first 1000 notes into consideration. The following equation is used to quantify similarity, where $E(a,b)$ denotes the edit distance between two pitch sequences.

$$s = 1 - \frac{E(a,b)}{1000} \tag{4.11}$$

We set the similarity threshold (t_{sim}) at 0.75. Although this is quite conservative, we found that this did not eliminate too many artifacts, while providing confidence that duplicate artifacts are not included in the dataset. Table 4.1 lists the size of each corpus after each preprocessing step.

Preprocessing Step	0	1	2
Wikiart	19052	19052	18874
Classical MIDI Archives	14724	12117	11943

Table 4.1: The corpus size after each preprocessing step

4.6.3 Data Representation

In order to avoid taking metadata, such as the title, composer, and author into consideration when computing the *NCD*, we do not use a binary representation of the MIDI files. Instead we create a representation which excludes irrelevant data. Since the velocity of MIDI note onsets is primarily based on the performer's interpretation of the composition, and in some cases may be set to a constant value if the MIDI file was created in a notation editor, we ignore this information. As a result, we represent a MIDI file as a sequence of onsets, offsets and time deltas. We represent onsets on the range $[0, 127]$, offsets on the range $[128, 255]$, and time deltas on the range $[256-)$. This results in a sequence of integers, which is then converted to a binary string before measuring the *NCD*. The representation used for images is much simpler. Each image is resized to have the shape 64×64, with three color channels (RGB), where each pixel is represented as an integer on the range $[0, 255]$.

4.6.4 Results

In Table 4.2 we present the results of 1000 trials, half of which have a ground truth of 0, and half which have a ground truth of 1, for a variety of corpora sizes. The *accuracy* (ACC), *true positive rate* (TPR), *false positive rate* (FPR), *true negative rate* (TNR), and *false negative rate* (FNR), are reported, using the formulas shown below, where n is the number of trials. *True positive* indicates trials in which the statistical test predicts 1 and the ground truth is also 1 ($\hat{g}(a, b) = 1 \wedge g(a, b) = 1$). Similarly, *true negative* indicates trials in which the statistical test predicts 0 and the ground truth is also 0 ($\hat{g}(a, b) = 0 \wedge g(a, b) = 0$). $\varepsilon = \varepsilon_I = \varepsilon_S$ denotes the equivalence range, which is normalized with respect to the length of $F = [f_i, i = 1, ..., n]$ and $G = [g_i, i = 1, ..., m]$ in Table 4.2. For example, if $\varepsilon = 0.1$ denotes an equivalence range of $(m + n) * 0.1$.

	Test	Corpus A		Corpus B		ACC	TPR	TNR	PPV	NPV	ε
		Size	Classes	Size	Classes						
WikiArt	p_{diff}	25	23	25	23	0.85	0.96	0.75	0.79	0.95	-
	p_{eqv}	25	23	25	23	0.78	0.78	0.77	0.78	0.77	0.15
	p_{diff}	50	23	50	23	0.92	0.97	0.87	0.88	0.96	-
	p_{eqv}	50	23	50	23	0.86	0.85	0.87	0.87	0.85	0.1
	p_{diff}	100	23	100	23	0.94	0.98	0.90	0.90	0.98	-
	p_{eqv}	100	23	100	23	0.92	0.94	0.90	0.90	0.93	0.075
	p_{diff}	50	23	100	23	0.82	0.98	0.67	0.75	0.97	-
	p_{eqv}	50	23	100	23	0.88	0.87	0.88	0.88	0.88	0.0875
Classical	p_{diff}	25	74	25	74	0.98	0.99	0.97	0.97	0.99	-
Archvies	p_{eqv}	25	74	25	74	0.92	0.88	0.95	0.94	0.89	0.15
	p_{diff}	50	37	50	37	0.99	1.00	0.99	0.99	1.00	-
	p_{eqv}	50	37	50	37	0.91	0.92	0.90	0.90	0.92	0.1
	p_{diff}	100	20	100	20	0.99	1.00	0.99	0.99	1.00	-
	p_{eqv}	100	20	100	20	0.93	0.94	0.91	0.92	0.94	0.075
	p_{diff}	50	37	100	20	0.85	1.00	0.75	0.77	0.99	-
	p_{eqv}	50	37	100	20	0.89	0.87	0.91	0.91	0.87	0.0875

Table 4.2: The results of 1000 randomized trials for each statistical test $(p_{\text{eqv}}, p_{\text{diff}})$ using a variety of corpora sizes.

$$ACC = \frac{\sum \text{True positive} + \sum \text{True negative}}{n} \tag{4.12}$$

$$TPR = \frac{2\sum \text{True positive}}{n} \tag{4.13}$$

$$TNR = \frac{2\sum \text{True negative}}{n} \tag{4.14}$$

$$PPV = \frac{\sum \text{True positive}}{\sum \text{Predicted positive}} \tag{4.15}$$

$$NPV = \frac{\sum \text{True positive}}{\sum \text{Predicted negative}} \tag{4.16}$$

A robust statistical test, will minimize the probability of type I error (α), incorrectly rejecting a true null hypothesis, and type II error (β), incorrectly rejecting a true alternative hypothesis. The power of a statistical test is $1 - \beta$, which is equivalent to the *TNR* with respect to the test for difference (p_{diff}), and the *TPR* with respect to the test for equivalence (p_{eqv}). Since we also must verify that the tests minimize type I error, we provide the *TPR* and *TNR* which are equivalent to statistical sensitivity, for p_{diff} and p_{eqv} respectively. For each trial, we perform 1000 permutations, as this is what Marozzi suggests when estimating the power of a permutation test [25].

4.7 Discussion

Given the degree of intra-corpus variation, and inter-corpus similarity, it is difficult to establish a ground truth for corpus comparison. In many cases, an artist or composer may explore several different sub-styles over the span of their career. Furthermore, artists and composers are often inspired by their colleagues, creating works that exhibit a greater than average degree of similarity. As a result, it would be unreasonable to expect extremely high values of accuracy. Nevertheless, according to Cohen, 0.80 is an adequate level for statistical power [9], which most of the tests surpass. Overall, the results of the experiment demonstrate that the proposed tests provide a robust measurement of the stylistic difference between two corpora.

We used different values for ε to account for the decrease in variability of w_A, w_B and $b_{A,B}$ as the size of the corpora increases. For example, if two paintings are randomly selected from the work of a single artist, in some cases, given the variability of that artist's work, the mutual information between these two paintings will be fairly low. In other cases, when both paintings are part of the same sub-style, the mutual information may be fairly high. However, as we increase the number of paintings selected, stylistic tendencies will start to emerge, and the amount of mutual information amongst the selected paintings will converge. As a result, w_A and w_B will decrease in variability as the size of the corpora is increased, which allows us to decrease the size of the equivalence interval by decreasing ε.

The results in Table 4.2 show two trends. On average the statistical tests performed better on MIDI than on images. There are two possible explanations for this; composers may have a more consistent style than artists, or the representation we used for images is not optimized for comparison. However, the fact that images were not preprocessed, as we simply resized each image and extracted the raw pixel values, demonstrates that *NCD* is capable of finding commonalities in the raw data. Secondly, the statistical tests perform better on larger corpora than smaller corpora, which is primarily the result of decreasing stylistic variability as the size increases.

Since the strings that are being compared were quite long, the *NCD* between two items was heavily skewed towards 1, as shown in Figure 4.1 and 4.2. Consequently, we do not suggest interpreting these values as interval, but rather as ordinal values. Despite the skew of these values, discrepancies between w_A, w_B, and $b_{A,B}$ can be quite pronounced.

4.8 Application

There are several ways in which the proposed tests could be used. In the most basic sense, the tests could be used to compare the source corpus (C) with a corpus of artifacts generated by the SI system (G). The magnitude of p_{eqv} can indicate how similar the two corpora are. In the case that $p_{\text{eqv}} >= \alpha$, the test for difference can be used to determine if there is a significant difference between the two corpora. In addition, it may be of particular interest to measure the similarity of $\hat{C}_s$ and $\hat{G}_s$, which denotes the projection of C and G into a lower dimensional feature space s. For example, in the music domain, one could use a representation that only contains rhythmic information, and another that only contains information about the harmonic progression, to gauge the degree to which the SI emulates the rhythm, and harmonic progressions which characterize C.

These tests could also be used to assess the CAN [13], which attempts to produce visual art in a style that is distinct from those it is trained on. In this scenario, we would have a set of corpora on which the CAN is trained ($S = C_i : i = 1, ..., n$), and for each $C_i \in S$ we would need to verify that $p_{\text{diff}} < \alpha$, using corrections for multiple hypothesis testing. Most importantly, since *NCD* operates on binary strings, these statistical tests are domain independent, as any digital data can be represented as a binary string.

4.9 Conclusion

Scientific progress is hindered in the absence of robust evaluation methodologies. This is an issue of particular contention in the field of computational creativity, as the subjective nature of assessments on creative artifacts can be problematic. In addition to issues of adequate domain knowledge, bias, and inter-rater reliability, the finite capacity of human participants limits the scalability of many evaluation approaches. This is a particular issue for SI systems, where the source corpus is often large, and the generated corpus is infinite. To address this issue, we propose CAEMSI for the evaluation of SI systems, providing compelling evidence that the statistical tests are reliable in two distinct domains. Future work involves further experimention with datasets from other domains, and the evaluation of generative systems with CAEMSI.

Bibliography

[1] Teresa M Amabile. “Social psychology of creativity: A consensual assessment technique.” In: *Journal of Personality and Social Psychology* 43.5 (1982), pp. 997–1013.

[2] Christopher Ariza. “The interrogator as critic: The turing test and the evaluation of generative music systems”. In: *Computer Music Journal* 33.2 (2009), pp. 48–70.

[3] Stefan Axelsson. “Using normalized compression distance for classifying file fragments”. In: *2010 International Conference on Availability, Reliability and Security*. IEEE. 2010, pp. 641–646.

[4] Anton Bardera, Miquel Feixas, Imma Boada, and Mateu Sbert. “Image registration by compression”. In: *Information Sciences* 180.7 (2010), pp. 1121–1133.

[5] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. *Deep Learning Techniques for Music Generation*. Computational Synthesis and Creative Systems. Springer International Publishing, 2019.

[6] Kevin Burns. “Computing the creativeness of amusing advertisements: A Bayesian model of Burma-Shave’s muse”. In: *AI EDAM* 29.1 (2015), pp. 109–128.

[7] Rudi Cilibrasi, Paul Vitányi, and Ronald De Wolf. “Algorithmic clustering of music”. In: *Proceedings of the Fourth International Conference onWeb Delivering of Music, 2004. EDELMUSIC 2004.* IEEE. 2004, pp. 110–117.

[8] Rudi Cilibrasi and Paul MB Vitányi. “Clustering by compression”. In: *IEEE Transactions on Information theory* 51.4 (2005), pp. 1523–1545.

[9] Jacob Cohen. *Statistical power analysis for the behavioral sciences*. Academic press, 2013.

[10] Darrell Conklin. “Discovery of distinctive patterns in music”. In: *Intelligent Data Analysis* 14.5 (2010), pp. 547–554.

[11] Ewa Dahlig. “Judgments of humans and machine authorship in real and artificial folksongs”. In: *Computing in musicology: a directory of research* 11 (1998), pp. 211–219.

[12] Tuomas Eerola, Tommi Himberg, Petri Toiviainen, and Jukka Louhivuori. “Perceived complexity of western and African folk melodies by western and African listeners”. In: *Psychology of Music* 34.3 (2006), pp. 337–371.

[13] Ahmed M. Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. “CAN: Creative Adversarial Networks, Generating "Art" by Learning About Styles and Deviating from Style Norms”. In: *Proceedings of the Eighth International Conference on Computational Creativity*. 2017, pp. 96–103.

[14] Celso França, Luıs Fabrıcio W Góes, Alvaro Amorim, Rodrigo Rocha, and Alysson Ribeiro Da Silva. “Regent-dependent creativity: A domain independent metric for the assessment of creative artifacts”. In: *Proceedings of the Seventh International Conference on Computational Creativity*. Citeseer. 2016, pp. 68–75.

[15] Ronald S Friedman and Christa L Taylor. “Exploring emotional responses to computationally-created music.” In: *Psychology of Aesthetics, Creativity, and the Arts* 8.1 (2014), p. 87.

[16] Albert Gatt and Emiel Krahmer. “Survey of the state of the art in natural language generation: Core tasks, applications and evaluation”. In: *Journal of Artificial Intelligence Research* 61 (2018), pp. 65–170.

[17] James C Kaufman, John Baer, and Jason C Cole. “Expertise, domains, and the consensual assessment technique”. In: *The Journal of creative behavior* 43.4 (2009), pp. 223–233.

[18] András Kocsor, Attila Kertész-Farkas, László Kaján, and Sándor Pongor. “Application of compression-based distance measures to protein sequence classification: a methodological study”. In: *Bioinformatics* 22.4 (2006), pp. 407–412.

[19] Carolyn Lamb, Daniel G Brown, and Charles LA Clarke. “Human Competence in Creativity Evaluation.” In: *ICCC*. 2015, pp. 102–109.

[20] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul MB Vitányi. “The similarity metric”. In: *IEEE transactions on Information Theory* 50.12 (2004), pp. 3250–3264.

[21] Ming Li and Ronan Sleep. “Genre Classification via a LZ78-Based String Kernel”. In: *ISMIR*. 2005, pp. 252–259.

[22] Feynman Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. “Automatic Stylistic Composition of Bach Chorales with Deep LSTM.” In: *Proceedings of the International Symposium on Music Information Retrieval*. 2017, pp. 449–456.

[23] Mary Lou Maher. “Evaluating creativity in humans, computers, and collectively intelligent systems”. In: *Proceedings of the 1st DESIRE Network Conference on Creativity and Innovation in Design*. Citeseer. 2010, pp. 22–28.

[24] Mary Lou Maher and Douglas H Fisher. “Using AI to evaluate creative designs”. In: *DS 73-1 Proceedings of the 2nd International Conference on Design Creativity Volume 1*. 2012.

[25] Marco Marozzi. “Some remarks about the number of permutations one should consider to perform a permutation test”. In: *Statistica* 64.1 (2004), pp. 193–201.

[26] David C Moffat and Martin Kelly. “An investigation into people’s bias against computational creativity in music composition”. In: *Assessment* 13.11 (2006), pp. 1–8.

[27] David Norton, Derrall Heath, and Dan Ventura. "Accounting for Bias in the Evaluation of Creative Computational Systems: An Assessment of DARCI." In: *ICCC*. 2015, pp. 31–38.

[28] Philippe Pasquier, Adam Burnett, Nicolas Gonzalez Thomas, James B Maxwell, Arne Eigenfeldt, and Tom Loughin. "Investigating listener bias against musical metacreativity". In: *Proceedings of the Seventh International Conference on Computational Creativity*. 2016, pp. 42–51.

[29] Philippe Pasquier, Arne Eigenfeldt, Oliver Bown, and Shlomo Dubnov. "An Introduction to Musical Metacreation". In: *Computer Entertainment* 14.2 (2017), pp. 3–17.

[30] Marcus T. Pearce. "The construction and evaluation of statistical models of melodic structure in music perception and composition". Dec. 2005.

[31] Marcus T. Pearce, David Meredith, and Geraint Wiggins. "Motivations and methodologies for automation of the compositional process". In: *Musicae Scientiae* 6.2 (2002), pp. 119–147.

[32] Marcus T. Pearce and Geraint A. Wiggins. "Evaluating cognitive models of musical composition". In: *Proceedings of the 4th international joint workshop on computational creativity*. Goldsmiths, University of London. 2007, pp. 73–80.

[33] Fortunato Pesarin, Luigi Salmaso, Eleonora Carrozzo, and Rosa Arboretti. "Union–intersection permutation solution for two-sample equivalence testing". In: *Statistics and Computing* 26.3 (2016), pp. 693–701.

[34] Graeme Ritchie. "Some empirical criteria for attributing creativity to a computer program". In: *Minds and Machines* 17.1 (2007), pp. 67–99.

[35] Samarendra Nath Roy. "On a heuristic method of test construction and its use in multivariate analysis". In: *The Annals of Mathematical Statistics* (1953), pp. 220–238.

[36] Markus Schedl, Arthur Flexer, and Julián Urbano. "The neglected user in music information retrieval research". In: *Journal of Intelligent Information Systems* 41.3 (2013), pp. 523–539.

[37] Sean L Simpson, Robert G Lyday, Satoru Hayasaka, Anthony P Marsh, and Paul J Laurienti. "A permutation testing framework to compare groups of brain networks". In: *Frontiers in computational neuroscience* 7 (2013), p. 171.

[38] Ray J Solomonoff. "A formal theory of inductive inference. Part I". In: *Information and control* 7.1 (1964), pp. 1–22.

[39] Nicolas Gonzalez Thomas, Philippe Pasquier, Arne Eigenfeldt, and James B Maxwell. "A Methodology for the Comparison of Melodic Generation Models Using Meta-Melo." In: *ISMIR*. 2013, pp. 561–566.

[40] Jaakko J Väyrynen, Tero Tapiovaara, Kimmo Kettunen, and Marcus Dobrinkat. "Normalized compression distance as an automatic MT evaluation metric". In: *Proceedings of MT* 25 (2010), pp. 21–22.

[41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[42] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. “MidiNet: A convolutional generative adversarial network for symbolic-domain music generation”. In: *arXiv preprint arXiv:1703.10847* (2017).

Chapter 5

Quantifying Musical Style: Ranking Symbolic Music based on Similarity to a Style

As published in Ens, J. & Pasquier, P. (2019). *Quantifying Musical Style: Ranking Symbolic Music based on Similarity to a Style.* ISMIR.

Abstract

Modelling human perception of musical similarity is critical for the evaluation of generative music systems, musicological research, and many Music Information Retrieval tasks. Although human similarity judgments are the gold standard, computational analysis is often preferable, since results are often easier to reproduce, and computational methods are much more scalable. Moreover, computation based approaches can be calculated quickly and on demand, which is a prerequisite for use with an online system. We propose StyleRank, a method to measure the similarity between a MIDI file and an arbitrary musical style delineated by a collection of MIDI files. MIDI files are encoded using a novel set of features and an embedding is learned using Random Forests. Experimental evidence demonstrates that StyleRank is highly correlated with human perception of stylistic similarity, and that it is precise enough to rank generated samples based on their similarity to the style of a corpus. In addition, similarity can be measured with respect to a single feature, allowing specific discrepancies between generated samples and a particular musical style to be identified.

Keywords: Generative Music; Evaluation; Machine Learning; Big Data

5.1 Introduction

Measuring musical similarity is a fundamental challenge, related to many tasks in Music Information Retrieval (MIR). In this paper, we focus on measuring the similarity between a MIDI file and an arbitrary musical style. In a musical context, the term style can refer to historical periods, composers, performers, sonic texture, emotion, and genre [9]. Here, we use the term style to denote the musical characteristics exhibited by a corpus $\mathcal{C} = \{C_1, ..., C_n\}$, as expressed by a feature set $\mathcal{F}$. Depending on the contents of $\mathcal{C}$, style may correspond to something as specific as a subset of a composer's work, as general as the entirety of Western Classical Music, or as personal as the musical preferences of an individual.

We propose StyleRank[1], a method for ranking MIDI files based on their similarity to a style delineated by $\mathcal{C}$. It can be used as a tool for musicological research, to evaluate Style Imitation (SI) systems, and to filter the output of an SI system. An SI system aims to generate music that exhibits the stylistic characteristics of $\mathcal{C}$ [28]. The primary contributions are as follows: a collection of novel features for symbolic music representation; an efficient MIDI feature extraction tool written in C++ with bindings in Python; a measure of similarity with respect to an arbitrary style delineated by $\mathcal{C}$; and two experiments demonstrating that this measure is robust, and highly correlated with human perception of stylistic similarity.

5.2 Motivations

There are several motivating factors for this research. In general, modelling human perception of musical similarity is of particular interest within the areas of Musicology, Music Cognition, and Music Theory [43]. Moreover, robust measures of musical similarity are critical for many MIR tasks, including database querying, music recommendation, and genre recognition. Although human perception is the gold standard for measuring musical similarity, natural human limitations place restrictions on the quantity and speed at which judgments can be collected, directly motivating automated measures of musical similarity.

More specifically, there are inherent challenges in designing a robust and reproducible listening experiment to evaluate SI systems. There are many variables which directly effect the quality of an experimental result, such as the number of participants, the listening environment, the sound equipment, and the number of samples selected for comparison. Even controlling for those variables, there is significant variability in how music is perceived, based on one's level of training [5] and musical background [16, 33, 14], which can result in a limited inter-rater agreement [35]. This is a particular issue, as it may hamper reproducibility and comparison with previously published results.

In most cases, sampling from an SI system is a stochastic process, and as a result, generated samples vary in quality. Developing a filtering process for generated material is a high priority

[1]The code is available at https://github.com/jeffreyjohnens/style_rank

concern, as low quality samples are undesirable when using a generative model in a production setting. Although measuring the log-likelihood of a sample can be useful as a proxy for quality, there are cases where log-likelihood significantly diverges from human perception. Theis et al. provide examples of generated images with high log-likelihood and extremely low quality [38]. With the exception of the feature-based statistical model of stylistic success proposed by Collins et al. [8], the authors are unaware of any pre-existing methods for ranking generated samples with respect to an arbitrary musical style.

5.3 Related Work

A wide variety of similarity measures have been developed to measure melodic [42], harmonic [32, 26, 10] and rhythmic similarity [39]. Many of these algorithms measure similarity by comparing two symbolic sequences [44]. Stylistic similarity, however, is rarely exhibited through sequence similarity, but rather through the repeated use of particular musical devices (i.e. melodic phrases, voice leading, and chord voicing) interspersed throughout the material [44]. In order to address this concern, approaches based on compression or pattern extraction have been proposed to measure similarity [22, 2, 6]. Since we aim to measure similarity with respect to $\mathcal{C}$, a more suitable approach will leverage information about the discriminative aspects of the entire corpus $\mathcal{C}$, rather than only taking two MIDI files into consideration.

In the context of SI system evaluation, the Turing Test [41] and the Consensual Assessment Technique [3] have been used to measure the stylistic similarity between generated artifacts $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_m\}$ and a particular style $\mathcal{C}$ [21, 29]. Objective measures have also been used to evaluate SI systems. Dong et al. measure the ratio of empty bars, pitch class diversity, note duration, rhythmic consistency, and tonal distance [11]. Trieu and Keller propose a variety of metrics ranging from rhythmic variety to harmonic consistency [40]. Since these metrics produce a single scalar value, it is easy to compare $\mathcal{C}$ and $\mathcal{G}$. However, these high-level metrics are likely only capable of measuring stylistic similarity in a very general sense. Sturm and Ben-Tal. plot distributions of meter, mode, number of tokens, pitch and pitch class for $\mathcal{C}$ and $\mathcal{G}$, but do not provide an automated method for analyzing discrepancies [37].

More comprehensive methodologies have been proposed, which involve computing all pairwise inter-set distances between samples in $\mathcal{C}$ and $\mathcal{G}$ ($D_{\mathcal{CG}} = [\texttt{dist}(c, g) : (c \in \mathcal{C}) \land (g \in \mathcal{G})]$), as well as all pairwise intra-set distances for samples within a set ($D_{\mathcal{GG}} = [\texttt{dist}(g_i, g_j) : (g_i \in \mathcal{G}) \land (g_j \in \mathcal{G}) \land (g_i \neq g_j)]$). [2] CAEMSI [18], a domain independent framework for the analysis of SI systems, provides a statistical method to test the null hypothesis $H_0 : (D_{\mathcal{GG}} \neq D_{\mathcal{CG}}) \lor (D_{\mathcal{CC}} \neq D_{\mathcal{CG}}) \lor (D_{\mathcal{CC}} \neq D_{\mathcal{GG}})$ against the alternative hypothesis $H_1 : D_{\mathcal{GG}} = D_{\mathcal{CC}} = D_{\mathcal{CG}}$. Yang and Lerch extract multi-dimensional features from each MIDI file [45]. For each feature, $D_{\mathcal{CG}}$

[2]Note that we adapt the set-builder notation to construct a list (e.g., $[i/2 : 0 \leq i < 4] = [0, 0, 1, 1]$), which unlike a set, may contain duplicate values.

and D_{CC} are constructed using Euclidean distance and smoothed using kernel density estimation [34, 27]. The distance between D_{CC} and D_{CG} is measured using (1) the area of overlap and (2) the Kullback–Leibler Divergence [19]. In contrast to both of these approaches, which involve evaluating the similarity between $\mathcal{G}$ and $\mathcal{C}$, StyleRank is optimized to evaluate the similarity of a single sample $g \in \mathcal{G}$ to $\mathcal{C}$.

5.4 Features

Although the features extracted by jSymbolic2 [24] are quite comprehensive, many features are high-level, and thus, ill-suited for the fine-grained distinctions that are necessary to rank stylistically similar MIDI files. For example, the `Chord Type Histogram` feature contains only 11 categories. In order to capture the complexity of the musical material being analyzed, we extract a variety of high-dimensional categorical distributions from a single MIDI file. A categorical distribution is a discrete probability distribution describing a random variable that has k possible distinct states. In what follows we adopt the following notation. Given a set x, $||x||$ denotes the number of elements in the set x, $\min(x)$ and $\max(x)$ denote the minimum and maximum element in x respectively, and x_i denotes the i^{th} element in x. $x \setminus y$ is the set difference between x and y, and $x \times y$ is the Cartesian product of x and y. $\ll$ indicates a left bitwise shift and $\gg$ indicates a right bitwise shift. $\&$, $\vee$, and $|$ refer to the bitwise AND, XOR, and OR operations, respectively.

5.4.1 Pitch Class Set Representations

In order to reduce the number of chords, we discard octave information and represent chords as pitch class sets, using a 12-bit integer to denote the presence or absence of a particular pitch class ($\text{C} = 0, \text{C\#} = 1, ..., \text{B} = 11$). For example, the C-major chord $\{60, 64, 67\}$ corresponds to the pitch class set $x = \{0, 4, 7\}$, which corresponds to the integer $\sum_{i=1}^{||x||}(1 \ll x_i) = 2^0 + 2^4 + 2^7 = 145$. Since there are 12 pitch classes, there are $2^{12} = 4096$ pitch class sets, which greatly reduces the possible number of chords. However, it is possible to further reduce this space if we create an equivalence class for all transpositionally equivalent pitch class sets. For example, the pitch class sets $\{0, 4, 7\}$ and $\{2, 5, 10\}$ are transpositionally equivalent, as both are major chords, the only difference being their root. This results in 352 distinct pitch class sets (PCD). Using Eq. (5.1c) a PCD can be calculated, where x is an 12-bit integer. Notably, pitch class sets are considered equivalent under the reversal operation when calculating the Forte number of a pitch class set [15]. Consequently, the pitch class sets $\{0, 4, 7\}$ and $\{0, 3, 7\}$ have the same Forte number, but correspond to different PCD's.

$$\texttt{rot}(x,n,i) = (x \ll i) \mid (x \gg (n-i)) \,\&\, (2^n-1) \tag{5.1a}$$

$$\texttt{reduce}(x,n) = \min(\{\texttt{rot}(x,n,i) : 0 \leq i < n\}) \tag{5.1b}$$

$$\texttt{pcd}(x) = \texttt{reduce}(x, 12) \tag{5.1c}$$

Alternatively, a pitch class set x can be represented as the set of scales which are supersets of x. Given a scale $\mathbb{S}$, let $\mathbb{S}_i = \{(s+i) \mod 12 : s \in \mathbb{S}\}$. The scale representation can be calculated with Eq. (5.2), where $\mathbb{S}^{\text{M}} = \{0,2,4,5,7,9,11\}$ and $\mathbb{S}^{\text{H}} = \{0,2,3,5,7,8,11\}$ denote the major and harmonic minor scales respectively. $\phi(\cdot)$ returns 1 if the predicate $\cdot$ is true and 0 otherwise.

$$\texttt{sc}(x) = (\sum_{i=1}^{12} \phi(x \subseteq \mathbb{S}_i^{\text{M}}) \ll i) + (\sum_{i=1}^{12} \phi(x \subseteq \mathbb{S}_i^{\text{H}}) \ll (12+i)) \tag{5.2}$$

5.4.2 Feature Definitions

Given a MIDI file M, for each note $n \in M$, $\texttt{ons}(n)$ returns the onset time of n in ticks, $\texttt{dur}(n)$ returns the duration of n in ticks, and $\texttt{pitch}(n)$ returns the pitch. An ordered set containing the unique onsets $O = \{\texttt{ons}(n) : n \in M\}$ is constructed, and the i^{th} chord is the set of notes $\mathbb{C}^i = \{n : (\texttt{ons}(n) \leq O_i) \wedge (\texttt{ons}(n) + \texttt{dur}(n) > O_i)\}$. $\texttt{isOns}(\mathbb{C}, n)$ and $\texttt{isTie}(\mathbb{C}, n)$ are functions that return 1 if n is an onset or a tie respectively, and 0 otherwise. The function $\texttt{pc}_i(\mathbb{C}, n)$ returns 1 if n corresponds to the pitch class i and 0 otherwise. In order to simplify the feature definitions, we use Eq. (5.3d), which accepts a chord $\mathbb{C}$ and a set of functions F, and only returns 1 if there is an element in X for which each $f \in F$ evaluates to 1. As a result, $\mathbf{I}(\mathbb{C}, \{\texttt{isOns}, \texttt{pc}_i\})$ is 1 if there is a note $n \in \mathbb{C}$ that is an onset and is equivalent to the pitch class i.

$$\texttt{pc}_i(\mathbb{C},n) = \begin{cases} 0, & \text{if } \texttt{pitch}(n) \mod 12 \equiv i \\ 1, & \text{otherwise} \end{cases} \tag{5.3a}$$

$$\texttt{isOns}(\mathbb{C},n) = \begin{cases} 0, & \text{if } \max(\{\texttt{ons}(n) : n \in \mathbb{C}\}) > \texttt{ons}(n) \\ 1, & \text{otherwise} \end{cases} \tag{5.3b}$$

$$\texttt{isTie}(\mathbb{C},n) = 1 - \texttt{isOns}(\mathbb{C},n) \tag{5.3c}$$

$$\mathbf{I}(\mathbb{C},F) = \begin{cases} 0, & \text{if } \max(\{\prod_{i=1}^{||F||} F_i(\mathbb{C},n) : n \in \mathbb{C}\}) < 1 \\ 1, & \text{otherwise} \end{cases} \tag{5.3d}$$

Table 5.1 provides formal definitions of all the features, where $\mathbb{C}^t$ denotes the t^{th} chord, $\mathbb{M}^t$ denotes the t^{th} melody pitch, $\mathbb{P}^t = \{\texttt{pitch}(n) : n \in \mathbb{C}^t\}$, $\mathbb{O}^t = \{\texttt{ons}(n) : n \in \mathbb{C}^t\}$, and $\mathbb{K}^t = \{\texttt{pitch}(n) : (n \in \mathbb{C}^t) \wedge \texttt{isOns}(n)\}$. $\texttt{popcount}(\cdot)$ is a function that counts the number of set bits in an integer, $\texttt{pc}(x) = x \mod 12$ and $\texttt{pcc}(x) = |(x \mod 12) - 6|$. Dissonance is calculated using Stolzenburg's periodicity function [36], which we refer to as $\texttt{stol}(\cdot)$. Let $\texttt{diss}(\mathbb{P}, \mathbb{T}) = \frac{1}{||\mathbb{T}||} \sum_{x \in \mathbb{T}} \texttt{stol}(\bar{\mathbb{P}}^x)$, where $\mathbb{P}$ and $\mathbb{T}$ are pitch sets, and $\bar{\mathbb{P}}^x = \{\mathbb{P}_i - x : \mathbb{P}_i \in \mathbb{P}\}$. $\texttt{voiceMotion}(\cdot)$ is a function that accepts two successive pitch sets $(\mathbb{P}^t, \mathbb{P}^{t+1})$ and returns an integer corresponding to the type of voice motion. $\texttt{tonnetzLength}(\cdot)$ is a function that accepts a pitch class set and returns the length of the shortest path through Tonnetz [25] vertices containing each pitch class.

Each function is calculated for all valid values of t, resulting in a categorical distribution with unsigned 64-bit integers as the categories. For example, given a standard 4-voice Bach chorale containing m chords, the function `ChordSize` is calculated for $0 \leq t < m-2$, producing a categorical distribution with the categories $\{0, 1, 2, 3, 4\}$. In some cases, we weight values by chord duration, denoted by a $\star$ in the table. In the case that a function returns a set of values (`IntervalDist`), we combine the returned sets to form the categorical distribution. Since the number of categories k grows exponentially large for some features (e.g., `ChordShape`), we restrict $k \leq 1000$ by ranking categories according to the number of samples they appear in, removing infrequently occurring categories.

5.4.3 Implementation

We implement the feature extraction tool in C++, using pybind11 [17] to create Python bindings. The Midifile library[3] is used to parse MIDI files.

5.5 Similarity Computation

In the most general sense, we are interested in measuring the similarity between a single MIDI file $\mathcal{X}$ and a corpus $\mathcal{C} = \{\mathcal{C}_1, ..., \mathcal{C}_n\}$. We represent each MIDI file by applying a non-empty set of feature transformations $\mathcal{F} = \{f_1, ..., f_k\}$, producing a set of categorical distributions for each MIDI file. For each $f_i \in \mathcal{F}$, we aim to measure the similarity between a single categorical distribution $f_i(\mathcal{X})$ and a set of categorical distributions $f_i(\mathcal{C}) = \{f_i(\mathcal{C}_1), ..., f_i(\mathcal{C}_n)\}$. Using a distance metric $\mathscr{D}$, the average similarity could be calculated $\frac{1}{n} \sum_{i=1}^{n} 1 - \mathscr{D}(f_i(\mathcal{C}_i), f_i(\mathcal{X}))$. However, this approach does not leverage information about the discriminative aspects of the entire corpus. The results in Experiment 1 demonstrate the deficiencies of this approach. Instead, we use Random Forests [7] to construct an embedding space before measuring the average similarity. Although neural networks are often ideal for learning embeddings, the time required to train k neural networks is prohibitive for an online system.

Decision trees are commonly used to model complex data. When used to classify data, each terminal node represents a discrete class label, and an arbitrary input is classified based on the terminal node it reaches. Using a trained Random Forest, an input can be represented based on the terminal node it reaches in each decision tree. Given a Random Forest containing N decision trees each with L terminal nodes, an input can be represented as a vector $v \in \{0, 1\}^{N \times L}$. To learn an embedding for

[3]https://midifile.sapp.org/

	Feature Name	Function	Description
Chord	ChordDissonance $\star$	$\lfloor \mathtt{diss}(\mathbb{K}^t, \mathbb{K}^t) \rfloor$	the dissonance of onsets based on periodicity [36]
	ChordDistinctDurationRatio	$(1 \ll \lVert\{\mathtt{dur}(n) : n \in \mathbb{C}^t\}\rVert) \mid 2^{\lVert\mathbb{C}^t\rVert}$	the ratio of distinct note durations to chord size
	ChordDuration	$\max(\mathbb{O}^{t+1}) - \max(\mathbb{O}^t)$	the duration of a chord
	ChordLowestInterval	$\min(\mathbb{P}^t \setminus \{\min(\mathbb{P}^t)\}) - \min(\mathbb{P}^t)$	the difference between the lowest two notes
	ChordOnset	$(\sum_{i=1}^{\lVert\mathbb{C}^t\rVert}(\mathtt{isOns}(\mathbb{C}_i^t) \ll (i-1))) \mid 2^{\lVert\mathbb{C}^t\rVert}$	an integer representing which notes are onsets
	ChordOnsetPCD $\star$	$\mathtt{pcd}(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\mathtt{isOns}, \mathtt{pc}_i\}) \ll i))$	distinct pitch class set excluding ties
	ChordOnsetRatio	$(1 \ll \sum_{n \in \mathbb{C}^t} \mathtt{isOns}(n)) \mid 2^{\lVert\mathbb{C}^t\rVert}$	the ratio of onsets to chord size
	ChordOnsetShape $\star$	$\sum_{i=1}^{\lVert\mathbb{C}^t\rVert}(\mathtt{isOns}(\mathbb{C}^t, \mathbb{C}_i^t) \ll (\mathbb{P}_i^t - \min(\mathbb{P}^t)))$	piano roll type representation of onset pitches
	ChordOnsetTiePCD $\star$	$\mathtt{pcd}(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\mathtt{isOns}, \mathtt{pc}_i\}) \ll i)) +$ $\mathtt{pcd}(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\mathtt{isTie}, \mathtt{pc}_i\}) \ll i)) \ll 12$	concatenated distinct pitch class set of onsets and distinct pitch class set of ties
	ChordOnsetTieReduced $\star$	$\mathtt{reduce}((\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\mathtt{isOns}, \mathtt{pc}_i\}) \ll i)) +$ $(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\mathtt{isTie}, \mathtt{pc}_i\}) \ll (12+i))))$	concatenated pitch class set of onsets and pitch class set of ties reduced using Eq. (5.1b)
	ChordPCD $\star$	$\mathtt{pcd}(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\mathtt{pc}_i\}) \ll i))$	distinct pitch class set
	ChordPCDWBass $\star$	$\mathtt{pcd}(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\mathtt{pc}_i\}) \ll i)) + 2^{12+\mathtt{pc}(\min(\mathbb{P}^t))}$	distinct pitch class set with bass pitch class
	ChordPCSizeRatio	$(1 \ll \lVert\{\mathtt{pc}(p) : p \in \mathbb{P}^t\}\rVert) \mid 2^{\lVert\mathbb{P}^t\rVert}$	the ratio of distinct pitch classes to chord size
	ChordRange (ϕ_1)	$\max(\mathbb{P}^t) - \min(\mathbb{P}^t)$	the range of pitches in a chord
	ChordShape $\star$	$\sum_{p \in \mathbb{P}^t}(1 \ll (p - \min(\mathbb{P}^t)))$	piano roll type representation of chord pitches
	ChordSize	$\lVert\mathbb{C}^t\rVert$	the number of notes in a chord
	ChordTonnetz $\star$	$\mathtt{tonnetzLength}(\{\mathtt{pc}(x) : x \in \mathbb{P}^t\})$	length of shortest path through Tonnetz [25] vertices
Chord Transition	ChordSizeNgram	$\lVert\mathbb{C}^t\rVert + (\lVert\mathbb{C}^{t+1}\rVert \ll 8) + (\lVert\mathbb{C}^{t+2}\rVert \ll 16)$	an n-gram of chord sizes ($n = 3$)
	ChordTranBassInterval	$\mathtt{pc}(\min(\mathbb{P}^{t+1}) - \min(\mathbb{P}^t))$	pitch class interval between two lowest notes
	ChordTranDissonance	$\lfloor \mathtt{diss}(\mathbb{P}^t, \mathbb{P}^{t+1}) \rfloor$	the dissonance of intervals based on periodicity [36]
	ChordTranDistance	$\lvert \min(\mathbb{P}^{t+1}) - \min(\mathbb{P}^t) \rvert + \lvert \max(\mathbb{P}^{t+1}) - \max(\mathbb{P}^t) \rvert$	approximated voice leading distance
	ChordTranOuter	$\mathtt{pc}(\phi_1(\mathbb{P}^t)) + (\mathtt{pc}(\phi_1(\mathbb{P}^{t+1})) \ll 8) +$ $(\mathtt{pc}(\min(\mathbb{P}^t) - \min(\mathbb{P}^{t+1})) \ll 16)$	pitch class transition using only the outer notes
	ChordTranPCD	$\mathtt{reduce}((\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\mathtt{pc}_i\}) \ll i)) +$ $(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^{t+1}, \{\mathtt{pc}_i\}) \ll (12+i))), 24)$	transition between distinct pitch class sets
	ChordTranRepeat	$(\prod_{n \in \mathbb{C}^t} \mathtt{isOns}(n))(\mathbb{P}^t = \mathbb{P}^{t+1})$	chord repetition with onsets
	ChordTranScaleDistance	$\mathtt{popcount}(\mathtt{sc}(\mathbb{P}^t) \vee \mathtt{sc}(\mathbb{P}^{t+1}))$	hamming distance between scale representations
	ChordTranScaleUnion	$\mathtt{popcount}(\mathtt{sc}(\mathbb{P}^t) \mid \mathtt{sc}(\mathbb{P}^{t+1}))$	the union between scale representations
	ChordTranVoiceMotion	$\mathtt{voiceMotion}(\mathbb{P}^t, \mathbb{P}^{t+1})$	type of voice motion (contrary, oblique, etc.)
Mel.	MelodyNgram	$\sum_{i=0}^{3}(\mathbb{M}_{t+i+1} - \mathbb{M}_{t+i} \mod 12) \ll 8i$	n-gram of melodic intervals ($n = 3$)
	MelodyPCD	$\mathtt{pcd}(\sum_{i=0}^{11} \mathbf{I}(\{\mathbb{M}^{t+i} : 0 \leq i < 5\}, \{\mathtt{pc}_i\}) \ll i)$	distinct pitch class of successive melody notes
Inter.	IntervalClassDist	$\{\mathtt{pcc}(p_i - p_j) : (p_j < p_i) \wedge (p_i, p_j \in \mathbb{P}^t \times \mathbb{P}^t)\}$	interval class for each combination of chord pitches
	IntervalDist	$\{\mathtt{pc}(p_i - p_j) : (p_j < p_i) \wedge (p_i, p_j \in \mathbb{P}^t \times \mathbb{P}^t)\}$	interval for each combination of chord pitches

Table 5.1: Definitions for Chord features, Chord Transition features, Melody features (Mel.), and Interval features (Inter.). The $\star$ symbol indicates that a categorical distribution is weighted by chord duration.

a single feature transformation $f_i \in \mathcal{F}$, we train a Random Forest to discriminate between a collection of items $f_i(\mathcal{G}) = \{f_i(\mathcal{G}_1), .., f_i(\mathcal{G}_m)\}$ and a corpus $f_i(\mathcal{C}) = \{f_i(\mathcal{C}_1), ..., f_i(\mathcal{C}_n)\}$. Concretely, each $f_i(\mathcal{G}_i) \in f_i(\mathcal{G})$ is given the label 0, and each $f_i(\mathcal{C}_i) \in f_i(\mathcal{C})$ is given the label 1. We refer to the vector produced for a sample $\mathcal{X}$ as $\mathbf{R}_{\mathcal{X}}^{\mathcal{G},\mathcal{C},f_i}$. Breiman measures the similarity of two vectors using the dot product [7]. In order to weight each feature transformation ($f_i \in \mathcal{F}$) equally, we use cosine similarity (Eq. (5.4a)), which is simply the normalized dot product. The similarity between $\mathcal{X}$ and

$\mathcal{C}$ with respect to a set of features $\mathcal{F}$ is computed using Eq. (5.4b), which produces a scalar value on the range $[0, 1]$.

$$\texttt{cos}(X, Y) = \frac{X \cdot Y}{\sqrt{\sum_{i=1}^{N} X_i^2}\sqrt{\sum_{i=1}^{N} Y_i^2}} \tag{5.4a}$$

$$S_{\mathcal{X}}^{\mathcal{G},\mathcal{C},\mathcal{F}} = \frac{1}{||\mathcal{C}||||\mathcal{F}||}\sum_{c\in\mathcal{C}}\sum_{f\in\mathcal{F}} \texttt{cos}(\mathbf{R}_{\mathcal{X}}^{\mathcal{G},\mathcal{C},f}, \mathbf{R}_{c}^{\mathcal{G},\mathcal{C},f}) \tag{5.4b}$$

5.6 Experiments

In the following experiments, we train a Random Forest [7] using the scikit-learn python module [31]. We set the maximum tree depth at 5, the number of trees to 500, and measure the quality of the split using entropy. The class weight is balanced to be robust against size discrepancies between $\mathcal{C}$ and $\mathcal{G}$.

5.6.1 Experiment 1 : Analytic Testing

We test StyleRank with styles delineated by a single composer, and by an entire genre, using the Classical Archives MIDI dataset[4]. In total there are 75 composers, and 6 musical genres. More details on the composition of the dataset can be found in the Appendix[5]. We keep only one MIDI file per composition. Each MIDI file is represented as a list of pitches, sorted lexicographically according to onset and pitch. To compare two pieces, the Levenshtein distance [20] is measured twice, once for the first 100 pitches in each piece, and once for the last 100 pitches. We eliminate pieces which have a Levenshtein distance less than 0.75, after normalizing the distance on the range $[0, 1]$. We choose this conservative value to ensure all duplicates are removed.

Given two styles $A = \{a_1, ..., a_m\}$ and $B = \{b_1, ..., b_n\}$, where $m = 2n$, let $\mathcal{C} = \{a_i : 1 \leq i \leq n\}$, $\mathcal{G}_A = \{a_i : n < i \leq 2n\}$, $\mathcal{G}_B = B$, and $\mathcal{G} = \mathcal{G}_A \cup \mathcal{G}_B$. By construction $\mathcal{G} \cap \mathcal{C} = \varnothing$. We train a Random Forest and compare two distributions $x = [S_g^{\mathcal{G},\mathcal{C},\mathcal{F}} : g \in \mathcal{G}_A]$ and $y = [S_g^{\mathcal{G},\mathcal{C},\mathcal{F}} : g \in \mathcal{G}_B]$, where $\mathcal{F}$ denotes the set of features described in Table 5.1. Ideally, each value in x should be larger than all values in y, since elements in $\mathcal{G}_A$ and $\mathcal{C}$ belong to the same style (A). However, depending on the specificity of the style, there may be some degree of overlap between A and B. In order to determine if there is a measurable difference between x and y we directly compare the means $(\bar{x} > \bar{y})$, and we calculate the p-value $(p^{\bar{x}>\bar{y}})$ for a One-Sided Mann-Whitney test [23] with the alternative hypothesis that $\bar{x} > \bar{y}$.

In cases where multiple statistical comparisons are performed, it is common practice to apply a correction to the raw p-values. The Bonferroni correction [12] is calculated by dividing the de-

[4]https://www.classicalarchives.com/midi.html

[5]https://github.com/jeffreyjohnens/style_rank/tree/master/appendix

		StyleRank				Cosine				Manhattan				Euclidean			
	size	μ	Sig	FDR	Bon	μ	Sig	FDR	Bon	μ	Sig	FDR	Bon	μ	Sig	FDR	Bon
Composer	10	0.963	0.86	0.725	0.0	0.837	0.624	0.381	0.0	0.879	0.662	0.413	0.0	0.827	0.565	0.28	0.0
	25	0.951	0.888	0.807	0.609	0.808	0.583	0.422	0.24	0.793	0.578	0.415	0.244	0.729	0.532	0.363	0.226
	50	0.926	0.905	0.873	0.78	0.705	0.559	0.454	0.333	0.751	0.599	0.468	0.34	0.717	0.565	0.428	0.3
	100	1.0	0.986	0.973	0.951	0.713	0.636	0.59	0.515	0.723	0.633	0.568	0.486	0.715	0.626	0.571	0.504
Genre	10	0.81	0.379	0.0	0.0	0.68	0.193	0.0	0.0	0.686	0.2	0.0	0.0	0.645	0.176	0.0	0.0
	25	0.867	0.578	0.376	0.198	0.729	0.348	0.084	0.038	0.74	0.374	0.053	0.021	0.691	0.298	0.06	0.022
	50	0.88	0.715	0.59	0.432	0.776	0.484	0.266	0.126	0.747	0.489	0.253	0.088	0.714	0.344	0.158	0.082
	100	0.927	0.847	0.774	0.671	0.766	0.555	0.406	0.265	0.755	0.566	0.44	0.284	0.785	0.462	0.269	0.178

Table 5.2: The normalized frequency over 1000 trials where $\bar{x} > \bar{y}$ (μ), $p^{\bar{x}>\bar{y}} < 0.05$ (Sig), $p^{\bar{x}>\bar{y}}$ is significant after applying the FDR correction (FDR), and $p^{\bar{x}>\bar{y}}$ is significant after applying the Bonferonni correction (Bon). Size denotes the size of the corpus $||\mathcal{C}|| = ||\mathcal{G}_A|| = ||\mathcal{G}_B||$.

sired level of significance ($\alpha = 0.05$) by the number of comparisons. The Benjamini–Yekutieli procedure [4] controls the false discovery rate under arbitrary dependence assumptions, and is less conservative than the Bonferroni correction. Given m null hypotheses and their corresponding p-values $P_1, ..., P_m$, the p-values are sorted in ascending order. For a given level of significance, in our case $\alpha = 0.05$, reject the null hypothesis for the first k values that satisfy $P_k \leq k\alpha/(m * c(m))$ where $c(m) = \sum_{i=1}^{m} 1/i$.

Table 5.2 shows the results of 1000 trials, reporting the percentage of trials where $\bar{x} > \bar{y}$, and the percentage of trials where $p^{\bar{x}>\bar{y}}$ is significant, applying no correction ($\alpha = 0.05$), the Benjamini–Yekutieli procedure (FDR), and the Bonferroni correction (Bon). We compare StyleRank against three distance measures, Cosine, Manhattan and Euclidean, replacing $S_g^{\mathcal{G},\mathcal{C},\mathcal{F}}$ with $\frac{1}{||\mathcal{C}||||\mathcal{F}||} \sum_{c \in \mathcal{C}} \sum_{f \in \mathcal{F}} 1 - \mathscr{D}(f(c), f(g))$.

5.6.2 Experiment 2: Congruity with Human Perception

In order to evaluate how well StyleRank correlates with human perception, we use data from the BachBot [21] experiment. In total, there were 5,967 participants, including 1329 novices, 2786 intermediate, 1341 advanced and 511 experts. Liang et al. generated 36 samples ($\mathcal{G}$) from a neural network trained on a collection of Bach Chorales ($\mathcal{C}$). Participants were asked to discriminate between a generated musical excerpt and an actual Bach chorale. They were each asked to complete 5 comparisons.

For each $g \in \mathcal{G}$, we count the number of times it was mistakenly classified as a Bach chorale N_g^{miss}, and the number of times it was correctly identified as computer generated N_g^{corr}. The raw count data can be found in the Appendix. We take the relative frequency of miss-classifications $T_g = N_g^{\text{miss}}/(N_g^{\text{miss}} + N_g^{\text{corr}})$ as an indication of how similar g is to the style of Bach's Chorales ($\mathcal{C}$). This results in $\binom{36}{2} = 630$ pairwise comparisons for which we have a ground truth ranking. Using a chi-square contingency test [30] we can measure the degree to which we are certain that there is a difference between two samples. We measure accuracy using Eq. (5.5b), where p_{ij} is the p-value for the chi-square contingency test comparing the counts for the i^{th} and j^{th} examples, $\phi(\cdot)$

	Novice				Intermediate			
	$\alpha = 5.0$	$\alpha = 0.5$	$\alpha = 0.05$	$\alpha = 0.005$	$\alpha = 5.0$	$\alpha = 0.5$	$\alpha = 0.05$	$\alpha = 0.005$
Random	.482 ± .025	.479 ± .031	.466 ± .044	.440 ± .062	.500 ± .023	.500 ± .026	.502 ± .033	.499 ± .037
jSymbolic	.471 ± .006	.463 ± .008	.472 ± .012	.491 ± .015	.478 ± .011	.474 ± .013	.467 ± .014	.456 ± .017
Loglik	.629 ± .000	.669 ± .000	.764 ± .000	.817 ± .000	.654 ± .000	.668 ± .000	.690 ± .000	.732 ± .000
StyleRank	.716 ± .001	.774 ± .002	.855 ± .004	.899 ± .005	.702 ± .002	.715 ± .002	.758 ± .002	.808 ± .002
	Advanced				Expert			
Random	.511 ± .010	.514 ± .013	.512 ± .017	.515 ± .019	.493 ± .019	.492 ± .025	.492 ± .032	.485 ± .038
jSymbolic	.481 ± .011	.480 ± .011	.470 ± .014	.474 ± .013	.452 ± .008	.449 ± .009	.482 ± .012	.464 ± .013
Loglik	.673 ± .000	.694 ± .000	.730 ± .000	.724 ± .000	.657 ± .000	.692 ± .000	.741 ± .000	.800 ± .000
StyleRank	.718 ± .001	.756 ± .001	.806 ± .002	.808 ± .002	.692 ± .002	.745 ± .003	.821 ± .004	.881 ± .005

Table 5.3: The accuracy of each model, calculated using Eq. (5.5b), with standard error calculated over 10 trials.

is a function returning 1 if the predicate $\cdot$ is true and 0 otherwise, and α denotes the threshold for significance.

$$f(x,y) = \begin{cases} 1, & \text{if } \phi(S_x^{\mathcal{G},\mathcal{C},\mathcal{F}} < S_y^{\mathcal{G},\mathcal{C},\mathcal{F}}) = \phi(T_x < T_y) \\ 0, & \text{otherwise} \end{cases} \tag{5.5a}$$

$$\texttt{acc}(\mathcal{G},\mathcal{C},\alpha) = \frac{\sum_{i=1}^{||\mathcal{G}||}\sum_{j=i+1}^{||\mathcal{G}||} f(\mathcal{G}_i,\mathcal{G}_j)\phi(p_{ij} < \alpha)}{\sum_{i=1}^{||\mathcal{G}||}\sum_{j=i+1}^{||\mathcal{G}||} \phi(p_{ij} < \alpha)} \tag{5.5b}$$

The results for Experiment 2 are presented in Table 5.3. We report the accuracy, calculated using Eq. (5.5b), for a random ranking (Random), StyleRank with the jSymbolic [24] features (jSymbolic), Log-likelihood (Loglik), and StyleRank. All the default features are extracted using jSymbolic, and features with zero standard deviation are removed. This results in a single feature vector with dimension of 453, for which we train a single Random Forest. Using the Performance RNN [13], which was trained with the same representation and data as the original BachBot, we evaluate the negative log-likelihood $\mathscr{L}_g$ of each of the generated examples (loglik). To calculate the accuracy we simply replace the term $S_X^{\mathcal{G},\mathcal{C},\mathcal{F}} < S_Y^{\mathcal{G},\mathcal{C},\mathcal{F}}$ with $\mathscr{L}_X < \mathscr{L}_Y$ in Eq. (5.5a).

5.7 Discussion

Collectively, the results of both experiments demonstrate that StyleRank is robust to corpora of varying sizes, and highly correlated with human perception of stylistic similarity. In the Appendix, we expand Experiment 1 to demonstrate that StyleRank's performance is robust, even when the number of distinct styles in $\mathcal{G}$ is increased. In Experiment 1, there is a large difference between raw distance measures and StyleRank. This highlights the limitations of the approach described by Yang and Lerch, which uses euclidean distance to measure the distance between feature vectors [45]. Although euclidean distance works well in low-dimensional settings, it does not scale well to high dimensions. In fact, it has been shown that Manhattan distance performs better than Euclidean distance in high dimensional settings [1], which we also see in our own experimental results. Understandably, there is a decrease in performance when analyzing styles delineated by genre, as these styles have more variance, and are less consistent than the work of a single composer. Overall, these results demonstrate that StyleRank can proficiently rank MIDI files with different styles.

The results for Experiment 2 demonstrate that StyleRank is capable of making fine-grained distinctions between MIDI files that correspond with human perception of stylistic similarity. It is worth noting that participants found it difficult to discriminate between generated and human-composed samples in the BachBot experiment, evidenced by the average classification accuracy of novice (0.57), intermediate (0.64), advanced (0.68), and expert (0.71) participants [21]. Based on our experimental results, the jSymbolic [24] feature set is no better at predicting rankings than a random model. This is likely due to the fact that high level features are not sufficiently discriminative for this task. In contrast to the jSymbolic feature set, our method involves full categorical distributions, which we believe are critical in measuring fine-grained differences. Importantly, there is a substantial difference between the accuracy of rankings based on log-likelihood and StyleRank. Interestingly, both log-likelihood and StyleRank best model high certainty ($\alpha = 0.005$) comparisons made by self identified novices. This may be an artifact of increased variance as the number of ground truth comparisons decreases as α increases.

It should be noted that participants in the BachBot experiment were not directly asked to rank samples according to their similarity to the style of Bach's chorales. We extrapolated a ranking from the number of times a sample was miss-classified, which is an indirect way of measuring stylistic similarity. However, since these rankings were based on a large sample size, we are confident that they are reflective of human perception.

5.8 Application

StyleRank can be used in a variety of settings. Importantly, we must note that there are no limitations on the composition of $\mathcal{G}$. For example, one could compare k different sets with $\mathcal{G} = \{\mathcal{G}_i^1, ..., \mathcal{G}_{n_1}^1, \mathcal{G}_1^2, ..., \mathcal{G}_{n_2}^2, ..., \mathcal{G}_1^k, ..., \mathcal{G}_{n_k}^k\}$. First of all, the method can be use to rank samples generated by an SI system, based on their similarity to $\mathcal{C}$. StyleRank can be used to filter highly dissimilar samples automatically. Filtering is as simple as taking the samples $g \in \mathcal{G}$ with a similarity $S_g^{\mathcal{G},\mathcal{C},\mathcal{F}}$ above some threshold, and discarding the rest. Secondly, StyleRank can be used to rank models. Given k models, let $\mathcal{G} = \{\mathcal{G}^1, ..., \mathcal{G}^k\} = \{\mathcal{G}_i^1, ..., \mathcal{G}_{n_1}^1, ..., \mathcal{G}_1^k, ..., \mathcal{G}_{n_k}^k\}$, where $\mathcal{G}^i$ denotes the set of samples generated by the i^{th} model. Then the distributions $x_i = [S_g^{\mathcal{G},\mathcal{C},\mathcal{F}} : g \in \mathcal{G}^i]$ can be compared using an appropriate statistical test. Third, the method can be used to isolate the specific features f that deviate from the style delineated by $\mathcal{C}$ by comparing the distributions $x_f = [S_g^{\mathcal{G},\mathcal{C},f} : g \in \mathcal{G}]$ for each f in a set of features $\mathcal{F}$. In addition, StyleRank can be used as a tool for musicologists to explore variations in style.

5.9 Conclusion

Quantifying musical stylistic similarity is a difficult task. We propose StyleRank, a method to rank individual MIDI files based on their similarity to an arbitrary style. Experimental evidence supports our approach, demonstrating that our method is robust, and is highly correlated with human perception of stylistic similarity. Future work involves applying this approach to other domains where SI systems are being developed. Additional features can be added to the current collection, in particular rhythm-based features, as the current collection is pitch-centric. Although we believe our experiments to be fairly comprehensive, continued validation of the proposed method on additional data is always beneficial.

Bibliography

[1] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. “On the surprising behavior of distance metrics in high dimensional space”. In: *International Conference on Database Theory*. 2001, pp. 420–434.

[2] Teppo E Ahonen, Kjell Lemström, and Simo Linkola. “Compression-based Similarity Measures in Symbolic, Polyphonic Music.” In: *ISMIR*. 2011, pp. 91–96.

[3] Teresa M Amabile. “Social psychology of creativity: A consensual assessment technique.” In: *Journal of Personality and Social Psychology* 43.5 (1982), pp. 997–1013.

[4] Yoav Benjamini and Daniel Yekutieli. “The control of the false discovery rate in multiple testing under dependency”. In: *The Annals of Statistics* 29.4 (2001), pp. 1165–1188.

[5] Mireille Besson, Daniele Schön, Sylvain Moreno, Andréia Santos, and Cyrille Magne. “Influence of musical expertise and musical training on pitch processing in music and language”. In: *Restorative Neurology and Neuroscience* 25.3-4 (2007), pp. 399–410.

[6] Peter Boot, Anja Volk, and W Bas de Haas. “Evaluating the role of repeated patterns in folk song classification and compression”. In: *Journal of New Music Research* 45.3 (2016), pp. 223–238.

[7] Leo Breiman. “Random forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32.

[8] Tom Collins, Robin Laney, Alistair Willis, and Paul H Garthwaite. “Developing and evaluating computational models of musical style”. In: *AI EDAM* 30.1 (2016), pp. 16–43.

[9] Roger B. Dannenberg. “Style in Music”. In: *The Structure of Style: Algorithmic Approaches to Understanding Manner and Meaning*. Ed. by Shlomo Argamon, Kevin Burns, and Shlomo Dubnov. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 45–57.

[10] W Bas De Haas, Frans Wiering, and Remco C Veltkamp. “A geometrical distance measure for determining the similarity of musical harmony”. In: *International Journal of Multimedia Information Retrieval* 2.3 (Sept. 2013), pp. 189–202.

[11] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018, pp. 34–41.

[12] Olive J. Dunn. “Multiple comparisons among means”. In: *Journal of the American Statistical Association* 56.293 (1961), pp. 52–64.

[13] Douglas Eck, Adam Roberts, Jesse Engel, Curtis Hawthorne, and Ian Simon. *Magenta*. Apr. 2019.

[14] Tuomas Eerola, Tommi Himberg, Petri Toiviainen, and Jukka Louhivuori. “Perceived complexity of western and African folk melodies by western and African listeners”. In: *Psychology of Music* 34.3 (2006), pp. 337–371.

[15] Allen Forte. *The Structure of Atonal Music*. Yale [paperbacks]. Yale University Press, 1973.

[16] Erin E Hannon and Sandra E Trehub. “Metrical categories in infancy and adulthood”. In: *Psychological Science* 16.1 (2005), pp. 48–55.

[17] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. *pybind11 – Seamless operability between C++11 and Python*. https://github.com/pybind/pybind11. 2017.

[18] **Jeff Ens** and Philippe Pasquier. “CAEMSI : A Cross-Domain Analytic Evaluation Methodology for Style Imitation”. In: *International Conference on Computational Creativity*. 2018, pp. 64–71.

[19] Solomon Kullback and Richard A Leibler. “On information and sufficiency”. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[20] Vladimir Levenshtein. “Binary codes capable of correcting deletions, insertions, and reversals”. In: *Soviet Physics-Doklady* 10.8 (1966), pp. 707–710.

[21] Feynman Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. “Automatic Stylistic Composition of Bach Chorales with Deep LSTM.” In: *Proceedings of the International Symposium on Music Information Retrieval*. 2017, pp. 449–456.

[22] Ning-Han Liu, Yi-Hung Wu, and Arbee LP Chen. “Efficient kNN search in polyphonic music databases using a lower bounding mechanism”. In: *Multimedia systems* 10.6 (2005), pp. 513–528.

[23] Henry B Mann and Donald R Whitney. “On a test of whether one of two random variables is stochastically larger than the other”. In: *The Annals of Mathematical Statistics* 18.1 (1947), pp. 50–60.

[24] Cory McKay, Julie Cumming, and Ichiro Fujinaga. “jSymbolic 2.2: Extracting features from symbolic music for use in musicological and MIR research”. In: *Proc. of the International Symp. on Music Information Retrieval*. 2018.

[25] Arthur Oettingen. *Harmoniesystem in dualer Entwickelung*. Dorpat: W. Glaser, 1866.

[26] Jean-François Paiement, Douglas Eck, and Samy Bengio. “A probabilistic model for chord progressions”. In: *Proc. of the International Symp. on Music Information Retrieval*. 2005, pp. 11–15.

[27] Emanuel Parzen. “On estimation of a probability density function and mode”. In: *The annals of mathematical statistics* 33.3 (1962), pp. 1065–1076.

[28] Philippe Pasquier, Arne Eigenfeldt, Oliver Bown, and Shlomo Dubnov. “An Introduction to Musical Metacreation”. In: *Computer Entertainment* 14.2 (2017), pp. 3–17.

[29] Marcus T. Pearce and Geraint A. Wiggins. “Evaluating cognitive models of musical composition”. In: *Proceedings of the 4th international joint workshop on computational creativity*. Goldsmiths, University of London. 2007, pp. 73–80.

[30] Karl Pearson. “On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling”. In: *Breakthroughs in Statistics: Methodology and Distribution*. Ed. by Samuel Kotz and Norman L. Johnson. New York, NY: Springer New York, 1992, pp. 11–28.

[31] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. “Scikit-learn: Machine learning in Python”. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[32] Jeremy Pickens and Tim Crawford. “Harmonic models for polyphonic music retrieval”. In: *Proc. of the International Conference on Information and Knowledge Management*. 2002, pp. 430–437.

[33] Jon B Prince, Mark A Schmuckler, and William F Thompson. “The effect of task and pitch structure on pitch-time interactions in music”. In: *Memory & Cognition* 37.3 (2009), pp. 368–381.

[34] Murray Rosenblatt. “Remarks on some nonparametric estimates of a density function”. In: *The Annals of Mathematical Statistics* (1956), pp. 832–837.

[35] Markus Schedl, Arthur Flexer, and Julián Urbano. “The neglected user in music information retrieval research”. In: *Journal of Intelligent Information Systems* 41.3 (2013), pp. 523–539.

[36] Frieder Stolzenburg. “Harmony perception by periodicity detection”. In: *Journal of Mathematics and Music* 9.3 (2015), pp. 215–238.

[37] Bob L. Sturm and Oded Ben-Tal. “Taking the models back to music practice: Evaluating generative transcription models built using deep learning”. In: *Journal of Creative Music Systems* 2.1 (2017), pp. 1–29.

[38] Lucas Theis, Aäron van den Oord, and Matthias Bethge. “A note on the evaluation of generative models”. In: *International Conference on Learning Representations*. arXiv:1511.01844. 2016.

[39] Godfried T. Toussaint. “A Comparison of Rhythmic Similarity Measures.” In: *Proc. of the International Symp. on Music Information Retrieval*. 2004, pp. 242–245.

[40] Nicolas Trieu and Robert Keller. “JazzGAN: Improvising with Generative Adversarial Networks”. In: *6th International Workshop on Musical Metacreation*. 2018.

[41] Alan M. Turing. “Computing Machinery and Intelligence”. In: *Mind* 59 (1950), pp. 433–460.

[42] Valerio Velardo, Mauro Vallati, and Steven Jan. “Symbolic melodic similarity: State of the art and future challenges”. In: *Computer Music Journal* 40.2 (2016), pp. 70–83.

[43] Anja Volk, Elaine Chew, Elizabeth Hellmuth Margulis, and Christina Anagnostopoulou. *Music similarity: Concepts, cognition and computation*. 2016.

[44] Anja Volk, W Bas de Haas, and Peter Van Kranenburg. “Towards modelling variation in music as foundation for similarity”. In: *Proceedings of the 12th International Conference on Music Perception and Cognition and the 8th Triennial Conference of the European Society for the Cognitive Sciences of Music*. School of Music Studies, Aristotle University of Thessaloniki. 2012.

[45] Li-Chia Yang and Alexander Lerch. “On the evaluation of generative models in music”. In: *Neural Computing and Applications* 1 (Nov. 2018), pp. 1–12.

Chapter 6

Improved Listening Experiment Design for Generative Systems

As published in Ens, J. & Pasquier, P. (2020). *Improved Listening Experiment Design for Generative Systems.* AIMC.

Abstract

Designing robust listening experiments is a critical component of research on generative music systems, as they are often the primary mechanism by which systems are bench-marked. However, the field lacks a set of guidelines for designing these types of experiments. In order to provide substantiated recommendations for experimental design, we examine the role of two parameters: the proportion of questions and the proportion of participants, both of which are measured relative to the total number of observations. Somewhat surprisingly, these parameters vary significantly from study to study, demonstrating a lack of consensus within the research community. Using experimental data collected from previous studies, we compare the power and reliability of various experimental designs, and arrive at guidelines regarding these proportions.

Keywords: Evaluation, Methodology, Generative Systems

6.1 Introduction

When evaluating a generative music system (audio or symbolic), human-based assessments are considered the gold standard. In most cases, participants are provided with one or more musical excerpts, and are asked to rate or rank the provided excerpts based on their quality. However, there are no generally accepted guidelines or recommendations for the design of these studies, which is directly evidenced by a high level of variance in experimental designs across studies published in recent years. We use experimental evidence and theoretical reasoning to critically evaluate the design of previously published experiments, and rationalize recommendations for improved experimental design.

We make the distinction between four different methodologies for quantitatively evaluating generative musical systems via a listening test. A modified Turing test [16] can take two forms, one where participants are asked whether a single musical excerpt is computer-generated or human-composed (I) [6, 15, 4], and another where participants select the human-composed musical excerpt from a pair of musical excerpts (II) [12]. Another approach (III) tasks participants with selecting the higher quality excerpt from a pair of musical excerpts [9, 8, 7, 14], where one or more of the excerpts is computer-generated. The final method involves computing the average rating for excerpts from each source (IV) [2, 3, 13]. Note that we use the term source here rather than generative system, as real data is often included as a condition in the experiment.

There are many factors which influence the outcome of a listening experiment. These include, the cultural background of the participant [5], the listening equipment used in the study, and the physical condition of the participant. However, many of these factors can be difficult for experimenters to control, especially when conducting a listening test via an online crowd-sourcing platform. Here we focus on two hyper-parameters which can be directly controlled by the experimenter, the proportion of questions and the proportion of participants. Consider an experiment $\mathcal{E} = \{(Q^{\gamma_1}, S^{\alpha_1}, R^1), ..., (Q^{\gamma_{n_{\text{obs}}}}, S^{\alpha_{n_{\text{obs}}}}, R^{n_{\text{obs}}})\}$ consisting of $n_{\texttt{obs}}$ observations, given a set of questions $Q = \{Q^1, ..., Q^{n_{\text{ques}}}\}$ and a set of participants $S = \{S^1, ..., S^{n_{\text{par}}}\}$. Note that a question is simply a set of musical excerpts sampled from one or more sources, from which a participant must formulate a response. Given $\mathcal{E}$, the proportion of participants is $n_{\texttt{par}}/n_{\texttt{obs}}$ and the proportion of questions is $n_{\texttt{ques}}/n_{\texttt{obs}}$. Although these hyper-parameters play a significant role, they have not been thoroughly scrutinized in this context.

6.2 Experimental Design

An experiment is comprised of factors, which are simply independent variables that are manipulated by the experimenter. There are two types of factors: fixed factors, which have a fixed number of levels that are of interest to the researcher; and random factors, where a random subset of the large number of possible levels that are of interest to the researcher are included in the experiment. Typically, in a listening experiment evaluating generative systems, there is one fixed factor, where each level is a different source (i.e. a generative system or real data). The participant factor, is a well-known random factor included in most experiments. Practical limitations place restrictions on the total number of levels (i.e. participants) that can be feasible included in the experiment, which forces the experimenter to randomly sample from the participant population of interest. In experiments that evaluate generative systems, there is another important random factor, the questions. It is clearly impossible to include all possible questions within an experiment, so we must settle for a random sample of questions.

Once we have established the factors within an experiment, it is necessary to determine the experiment design, which specifies the relationship between factors. Pairs of factors can be crossed or nested. If two factors are crossed, every level of one factor co-occurs with every level the other factor. If factors are nested, each level of a factor co-occurs with only one level of the other. For example, consider a methodology I experiment conducted with two participants (S^1, S^2), comparing two sources (M^1, M^2), where 2 excerpts $(e_1^{M^k}, e_2^{M^k})$ are generated from each source M^k. Since methodology I asks participants to listen to a single excerpt and predict whether it was computer-generated or human-composed, we have two unique questions $(Q_1^{M^k}, Q_2^{M^k})$ per source, where each question consists of a single musical excerpt. Here, the question factor is nested within the source factor, as each question $Q_i^{M^k}$ is unique to the source M^k. Note that this is the case for all listening experiments evaluating generative systems, since it is exceedingly rare to sample the same question from two different sources. If each of the 4 questions are shown to each participant, then the participant factor would be crossed with the question factor, as each participant-excerpt combination $(S^i, e_j^{M^k})$ is part of the experiment.

There are three common experimental designs: crossed-question, partially-crossed-question and nested-question. A crossed-question design, shown in Figure 1a, exposes each participant to the same set of questions. A nested-question design, shown in Figure 1c, nests questions within participants, so that each participant is exposed to a different set of questions. It is also possible to employ a partially-crossed-question design, shown in Figure 1b, where the set of questions that each participant is exposed to is randomly drawn from a set of questions. As a result, participants will sometimes be exposed to the same question. Although 6 observations are collected in each of the experimental designs shown in Figure 1, the sample size of the question random factor varies. Consequently, the proportion of questions is smallest for a crossed-question design $(\frac{n_{\text{ques}}}{n_{\text{obs}}} = \frac{2}{6})$ and largest for a nested-question design $(\frac{n_{\text{ques}}}{n_{\text{obs}}} = \frac{6}{6})$.

For purposes of conceptual clarity, we only consider the case where an experiment consists of $\leq$ 2 sources, as this is an atomic unit that larger experiments are easily factored into. For example, consider the paired listening experiment presented in the Music Transformer paper [9], which compares four sources (Music Transformer, Transformer, LSTM, and the Maestro dataset [7]) using methodology III. This can be factored into $6 = \binom{4}{2}$ distinct sub-experiments corresponding to each possible pair of sources. Clearly, if we take steps to improve each sub-experiment, it will have a positive effect on the experiment as a whole.

	$Q_1^{M_1}$	$Q_2^{M_2}$
S^1	x	x
S^2	x	x
S^3	x	x

(a)

	$Q_1^{M_1}$	$Q_2^{M_1}$	$Q_3^{M_2}$	$Q_4^{M_2}$
S^1	x	-	x	-
S^2	-	x	x	-
S^3	-	x	-	x

(b)

	$Q_1^{M_1}$	$Q_2^{M_1}$	$Q_3^{M_1}$	$Q_4^{M_2}$	$Q_5^{M_2}$	$Q_6^{M_2}$
S^1	x	-	-	x	-	-
S^2	-	x	-	-	x	-
S^3	-	-	x	-	-	x

(c)

Figure 6.1: Three different experimental designs: crossed-question (a), partially-crossed-question (b) and nested-question (c). The cells with x denote the observations that are collected.

6.3 Motivation

There are several motivating factors for this research. First and foremost, without robust experimental design, any claims based on the experimental results are weakened, and in the extreme case completely invalid. Secondly, there are currently no standard recommendations for experimental design, which results significant discrepancies between studies. In Figure 6.2, we plot the proportion of questions ($\frac{n_{\texttt{ques}}}{n_{\texttt{obs}}}$), the proportion of participants ($\frac{n_{\texttt{par}}}{n_{\texttt{obs}}}$), the experimental design, and the methodology for several recent listening experiments for which the relevant information was available. Of particular concern, is the fact that the proportion of questions, and experiment design vary significantly across experiments, indicating a lack of consensus amongst the research community. Finally, given the high costs of conducting a study, it is essential that the studies produce accurate results and are implemented to make efficient use of the allocated resources.

6.4 Experiment 1 : Calculating Experimental Power

A typical approach to evaluate an experimental design is to calculate the power, which is simply the inverse of the probability of Type II error. In order to calculate the power of an experiment, there are two factors which must be considered: the variance components, and the sample size [10]. Note that in our case, there is not a single sample size, but rather a sample size for the participant random factor, and a sample size for the question random factor. To explore the differences between nested-question and crossed-question experiment designs, we conduct a parameter sweep for the number of participants, and the number of questions per participants, calculating the power for each pair of parameters. We use power calculations designed for experiments with two random factors [17], and compute the variance components from a previous experiment [2], which featured a crossed-question design. Since power calculations are not available for partially-crossed designs we can not

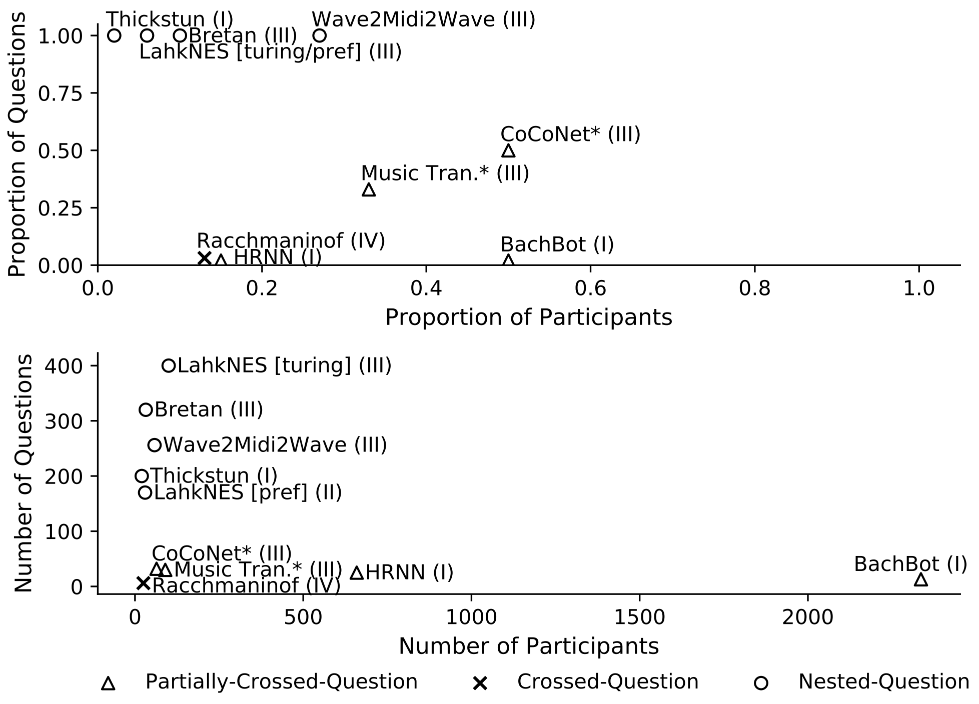


Figure 6.2: The experimental designs employed in recent listening studies for generative systems. Stars indicate that the number/proportion of participants could not be calculated exactly.

explicitly explore this experiment design here. We deliberately set the x axis of Figure 6.3 to be the number of questions per participant, rather than the total number of questions, so that the power at each (x,y) coordinate can be directly compared, as the total number of observations is equivalent for each experimental design.

The results in Figure 6.3 demonstrate that nested designs are uniformly more powerful than crossed designs, as we get an average of 2.3 times more power when using a nested experiment with the same number of total observations. The reason for this is rather straightforward, as a nested-question experiment design can make use of $n_{\texttt{obs}}$ unique questions, while in a crossed design we are restricted to $\frac{n_{\texttt{obs}}}{n_{\texttt{par}}}$ unique questions. Provided that variance components related to the question factor are non-zero and $n_{\texttt{par}} \geq 1$, nested-question experiments will always be more powerful than crossed question experiments, as they increase the sample size for the question factor by a factor of $n_{\texttt{par}}$ [10].

The dashed lines in Figure 3a show the different possible combinations of participants $(n_{\texttt{par}})$ and questions per participant given a constant number of observations $(n_{\texttt{obs}})$. This reveals that the power decreases when we decrease the proportion of participants $\frac{n_{\texttt{par}}}{n_{\texttt{obs}}}$, while holding the proportion of questions constant $(\frac{n_{\texttt{ques}}}{n_{\texttt{obs}}} = 1)$. Note that we cannot observe this same effect in Figure 3b, since changes to the proportion of participants are confounded with changes to the proportion of questions. The same type of effect can be observed in a nested-participant experiment design, where

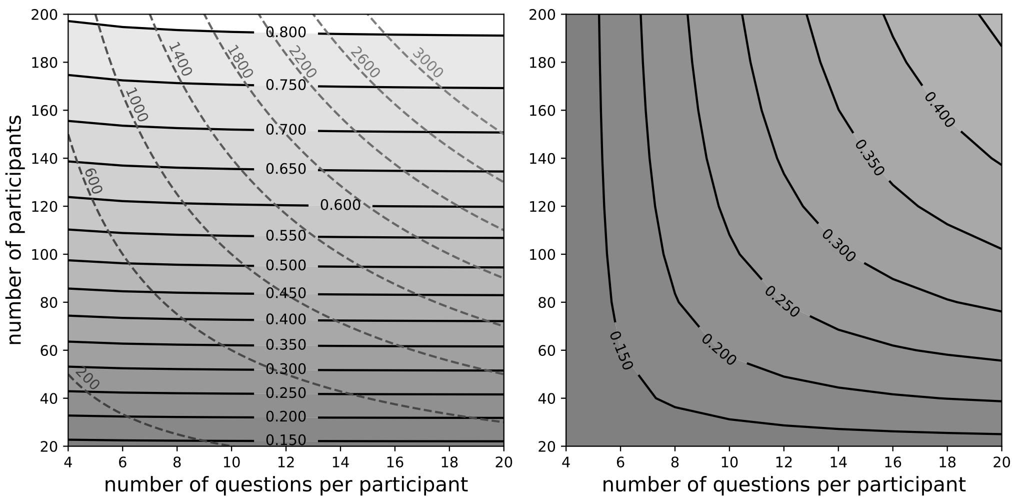


Figure 6.3: Power simulation for nested-question (left) and crossed-question (right) experimental designs using variance components estimated from Collins' study. Each dashed line in left plot illustrates the possible combinations of $n_{\texttt{par}}$ and questions per participant given a constant number of observations ($n_{\texttt{obs}}$).

the power decreases when the proportion of questions decreases. However, this type of experiment design is highly impractical as it requires collecting a single response from each participant.

We can also observe that in a crossed design, there is little advantage to increasing the number of participants or increasing the number of questions per participant separately. Power mainly increases when the number of participants and the number of questions per participant are increased together. Furthermore, it is possible to reach a point when adding an additional participant has no effect on the power at all, since the contour lines eventually become almost vertical. In contrast, when using a nested design most of the gains come from adding participants, since this effectively increases the total number of questions in the experiment, as the questions at each participant level are unique. The efficiency of the nested-question experiment design, is that it allows for both the sample size for participants and questions to be increased simultaneously. Collectively, these results demonstrate that crossed-question experiments are under-powered, and that decreasing the proportion of participants or the proportion of questions decreases the power.

6.5 Experiment 2 : Simulating Inter-Experiment Variance

In this experiment, we aim to measure inter-experiment variability, quantifying the reliability of different experimental designs. Formally, given an experiment $\mathcal{E}$, let $\psi^k_{n_{\texttt{par}},n_{\texttt{ques}}}$ denote the result (i.e. the average score for a source) for a randomly sampled subset of $\mathcal{E}$, containing k observations, $n_{\texttt{par}}$ participants, and $n_{\texttt{ques}}$ questions, where each each observation in $\psi^k_{n_{\texttt{par}},n_{\texttt{ques}}}$ involves the same source(s). To observe the difference between two experimental designs (α and β), we compute $\psi^k_{n^{\alpha}_{\texttt{par}},n^{\alpha}_{\texttt{ques}}}$ and $\psi^k_{n^{\beta}_{\texttt{par}},n^{\beta}_{\texttt{ques}}}$ r times, resulting in the sets Ψ^{α} and Ψ^{β}. Then we use Levene's test [11]

data source	k	$n^{\alpha}_{\texttt{par}}$	$n^{\alpha}_{\texttt{ques}}$	$n^{\beta}_{\texttt{par}}$	$n^{\beta}_{\texttt{ques}}$	proportion of significant trials
LahkNES [pref]	10	5	10	10	10	1.00
LahkNES [turing]	10	5	10	10	10	1.00
BachBot	10	10	2	10	10	1.00
Racchmaninoff	9	3	3	9	3	.95

Table 6.1: The proportion of trials where Ψ^{α} exhibits more variance than Ψ^{β}.

to determine if the variance of Ψ^{α} and Ψ^{β} differs significantly. The entire procedure is repeated 100 times with $r = 50000$, producing 100 p-values. In order to be sure that our results are simply not an artifact of the sub-experiment sampling procedure, we also conduct the same procedure using a version of the data where the responses have been randomly sampled from a uniform distribution, counting the proportion of times that $\sigma(\Psi^{\alpha}) < \sigma(\Psi^{\beta})$, where $\sigma(\Psi^{i})$ denotes the variance of the set Ψ^{i}. If the sub-experiment sampling procedure has a significant effect on the outcome, we would expect this proportion to vary significantly from 0.5, a hypothesis which can be tested using the Binomial test.

We use the experimental results provided by the authors of following four listening experiments: BachBot [12], Wave2Midi2Wave [7], LahkNES [4], and Racchmaninoff [2]. Although we contacted the authors of 15 different studies, we only received experimental results from the four listed above. Note that the experimental design of the original experiments will place inherent limitations on the types of simulations that we can conduct. In the BachBot study, each participant is presented with two different questions, randomly selected from a pool of 13 questions, resulting in a partially-crossed-question design. With this data, we can simulate a partially-crossed-question design $\psi^{10}_{10,2}$ and a nested-question design $\psi^{10}_{10,10}$. In the Wave2Midi2Wave and LahkNES studies, there are almost no duplicate questions in the entire experiment, which only allows us to manipulate the proportion of participants. We simulate two nested-question designs: $\psi^{10}_{5,10}$, and $\psi^{10}_{10,10}$. Using the Racchmaninoff data, we can simulate a crossed-question design $\psi^{9}_{3,3}$ and a partially-crossed-question design $\psi^{9}_{9,3}$. For each comparison, the proportion of significant results after applying the false discovery rate correction [1] is shown in Table 6.1. The Binomial test for the Wave2Midi2Wave simulations was significant, indicating that the sub-experiment sampling procedure biased the result, so this simulation was excluded from the results. In all other cases, the Binomial test was insignificant. Collectively, the results demonstrate that increasing the proportion of participants or questions decreases the inter-experiment variance, confirming the theoretical results presented in experiment 1.

6.6 Discussion and Recommendations

In addition to considering the power and reliability of a particular experimental design, it is also worth taking the end-point of the experiment into account. In most cases, the end-point of a listening study for the evaluation of generative systems is an average score for each system. In contrast, the endpoint of an experiment measuring the valence and arousal of audio clips, is the average valence and arousal for each audio clip. There is a subtle difference between these two types of experiments. In the first experiment, audio excerpts are a random factor, where we take a random sample from the entire population of possible generated excerpts. In the second experiment, audio excerpts are a fixed factor, where we are interested only in the levels contained within the experiment. We do not expect that the results for one particular audio excerpt will generalize to another audio excerpt in the second experiment. As a result, it makes sense to collect multiple observations from multiple participants for each audio excerpt, as we need the average response for each excerpt to be reflective of how the entire participant population feels about that excerpt. However, when conducting a prototypical listening experiment for generative systems, we care about what the entire population thinks of each source, not the individual audio excerpts. To make matters worse, our experiments demonstrated that collecting multiple observations for a single audio excerpt actually makes the results we actually care about less reliable and the experiment as a whole less powerful, as the size of the random sample for audio excerpts representing each source is unnecessarily reduced.

This is not to say that collecting multiple observations for a single audio excerpt is always wasteful. In fact, a crossed-question experiment was necessary for calculating the variance components used our simulations. Furthermore, in cases where inter-rater agreement is the endpoint of an experiment, it is necessary for $\frac{n_{\text{ques}}}{n_{\text{obs}}} < 1$. However, most listening experiments for generative systems do not measure inter-rater agreement. Ultimately, it is absolutely essential that the experiment design matches the goals of the research question, otherwise we often end up needlessly sacrificing power and reliability in our experiments. For those who are conducting a prototypical listening experiment for generative systems, we offer the following advice. Since resources (i.e. time and money) are finite, we will assume that a fixed number of observations $(n_{\texttt{obs}})$ can be collected, irrespective of the experiment design. As our experimental results demonstrate that the sample size of the question and participant random factors have a significant effect on the power and reliability of the experiment, an ideal experimental design will maximize $\frac{n_{\text{ques}}}{n_{\text{obs}}}$ and $\frac{n_{\texttt{par}}}{n_{\text{obs}}}$. First and foremost, this means it is essential to avoid crossed-question experimental designs, as they reduce $\frac{n_{\text{ques}}}{n_{\text{obs}}}$ by a factor of $n_{\texttt{par}}$. In most cases, there are relatively few barriers to selecting a nested-question design $(\frac{n_{\text{ques}}}{n_{\text{obs}}} = 1)$ or a partially-crossed-question design with a large proportion of questions, as sampling from most models is cheap. In fact, we have seen this experimental design employed in several listening studies [4, 15, 7]. However, there are many listening studies which feature a small proportion of questions, needlessly sacrificing power and reliability. Although our results demonstrate that collecting each response from a unique participant $(\frac{n_{\texttt{par}}}{n_{\text{obs}}} = 1)$ would be optimal, this may not be practical, as there are costs associated with obtaining each participant. Fortunately, most experiments do a good job balancing the proportion of participants, collecting a modest amount of responses from each participant.

6.7 Conclusion

We have examined two critical parameters for the experimental design of listening studies: the proportion of questions $\frac{n_{\texttt{ques}}}{n_{\texttt{obs}}}$, and the proportion of participants $\frac{n_{\texttt{par}}}{n_{\texttt{obs}}}$. Through experimentation we demonstrated that when $\frac{n_{\texttt{ques}}}{n_{\texttt{obs}}} < 1$ or $\frac{n_{\texttt{par}}}{n_{\texttt{obs}}} < 1$, the power and reliability of the experiment are reduced. Since listening studies are a fundamental aspect of research involving generative systems, and a consensus on best practices for listening experiment design has yet to emerge, these recommendations will undoubtedly be a useful reference point for future research.

Bibliography

[1] Yoav Benjamini and Daniel Yekutieli. “The control of the false discovery rate in multiple testing under dependency”. In: *Annals of statistics* (2001), pp. 1165–1188.

[2] Tom Collins and Robin Laney. “Computer-generated stylistic compositions with long-term repetitive and phrasal structure”. In: *Journal of Creative Music Systems* 1.2 (2017).

[3] Tom Collins, Robin Laney, Alistair Willis, and Paul H Garthwaite. “Developing and evaluating computational models of musical style”. In: *AI EDAM* 30.1 (2016), pp. 16–43.

[4] Chris Donahue, Huanru Henry Mao, Yiting Ethan Li, Garrison W Cottrell, and Julian McAuley. “LakhNES: Improving multi-instrumental music generation with cross-domain pre-training”. In: *Proc. of the 20th International Society for Music Information Retrieval Conference*. 2019, pp. 685–692.

[5] Tuomas Eerola, Tommi Himberg, Petri Toiviainen, and Jukka Louhivuori. “Perceived complexity of western and African folk melodies by western and African listeners”. In: *Psychology of Music* 34.3 (2006), pp. 337–371.

[6] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. “Deepbach: a steerable model for bach chorales generation”. In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 1362–1371.

[7] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse H. Engel, and Douglas Eck. “Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset”. In: *7th International Conference on Learning Representations*. 2019.

[8] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron C. Courville, and Douglas Eck. “Counterpoint by Convolution”. In: *Proceedings of the 18th International Society for Music Information*. 2017, pp. 211–218.

[9] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. “Music Transformer: Generating Music with Long-Term Structure”. In: *7th International Conference on Learning Representations*. 2019.

[10] Charles M Judd, Jacob Westfall, and David A Kenny. “Experiments with more than one random factor: Designs, analytic models, and statistical power”. In: *Annual Review of Psychology* 68 (2017), pp. 601–625.

[11] Howard Levene. “Contributions to probability and statistics”. In: *Essays in honor of Harold Hotelling* (1960), pp. 278–292.

[12] Feynman Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. “Automatic Stylistic Composition of Bach Chorales with Deep LSTM.” In: *Proceedings of the International Symposium on Music Information Retrieval*. 2017, pp. 449–456.

[13] Marcus T. Pearce and Geraint A. Wiggins. “Evaluating cognitive models of musical composition”. In: *Proceedings of the 4th international joint workshop on computational creativity*. Goldsmiths, University of London. 2007, pp. 73–80.

[14] Adam Roberts, Jesse H. Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. “A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music”. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 4361–4370.

[15] John Thickstun, Zaid Harchaoui, Dean P Foster, and Sham M Kakade. “Coupled Recurrent Models for Polyphonic Music Composition”. In: *Proc. of the 20th international society for music information retreival conference*. 2018, pp. 311–318.

[16] Alan M. Turing. “Computing machinery and intelligence”. In: *Parsing the Turing Test*. Springer, 2009, pp. 23–65.

[17] Jacob Westfall, David A Kenny, and Charles M Judd. “Statistical power and optimal design in experiments in which samples of participants respond to samples of stimuli.” In: *Journal of Experimental Psychology: General* 143.5 (2014).

Chapter 7

The Multi-Track Music Machine: A Generative System Designed for Co-Creative Music Composition

Abstract

We propose the Multi-Track Music Machine (MMM), a generative system based on the Transformer architecture that is designed to support co-creative music composition workflows. MMM supports the infilling of musical material on the track and bar level, and can condition generation on particular attributes including: instrument type, note density, polyphony level, and note duration. In order to integrate these features, we employ a different type of representation for musical material, creating a time-ordered sequence of musical events for each track and concatenating several tracks into a single sequence, rather than using a single time-ordered sequence where the musical events corresponding to different tracks are interleaved. We present experimental results which demonstrate that MMM is able to consistently avoid duplicating the musical material it was trained on, generate music that is stylistically similar to the training dataset, and that attribute controls can be employed to enforce various constraints on the generated material. We also outline several real world applications of MMM , including the production of musical albums, and collaborations with industry partners that explore integrating MMM into real-world products.

Keywords: Generative Music; Evaluation; Machine Learning; Big Data

	Input Specifications					Generation Methods	
System	Tracks	Instruments	Fixed Schema	Drums	Polyphony	Infill	Attr. Control
MMM	$\star$	128	-	x	x	x	Section 7.4
MuseNet [16]	10	10	-	x	x	-	x
MuseGAN [8]	4	4	x	x	x	-	-
LahkNES [7]	4	4	x	x	-	-	-
CoCoNet [9]	4	4	x	-	-	x	-
MusicVae [18]	3	3	x	x	-	-	-
SketchNet [4]	1	-	x	-	-	x	x
[15, 12]	1	-	x	-	-	x	-
[3, 5]	1	-	x	-	x	x	-
[20, 22, 21]	1	-	x	-	x	-	x

Table 7.1: A summary of the input specifications and generation methods of recently published generative music systems, where - indicates the absence of a particular feature. The $\star$ indicates that MMM does not have an explicit track limit.

7.1 Introduction

Research involving generative music systems has focused on modelling musical material as an end-goal, rather than on the affordances of such systems in practical scenarios [19]. As a result, there has been a focus on developing novel architectures and demonstrating that music generated with these architectures is of comparable quality to human-composed music, often via a listening test. Although this is a necessary first step, as systems must be capable of generating compelling material before they can be useful in a practical context, given the impressive capabilities of the Transformer-based models in the music domain [7, 10], we shift our focus to increasing the affordances of a Transformer-based system. Our primary contribution is the Multi-Track Music Machine (MMM), which utilizes a novel representation for multi-track musical material, resulting in an expressive and steerable generative system. We discuss ongoing real-world usage of MMM and provide quantitative evidence demonstrating that MMM: produces original variations without duplicating the training data; generates musical material that retains the stylistic characteristics of the training data; and that attribute control methods are an effective way to steer generation.

7.2 Comparison to Related Work

Given our interest in developing a system which is well-suited to co-creative music composition, it is worth identifying different factors which enhance the real-world usability of a generative music system. We consider two main categories: input specifications, which place restrictions on the musical material that can be processed by the system; and generation methods, which in some way augment the interaction between the user and the generative system.

7.2.1 Input Specifications

With regards to input specifications, we consider the number of tracks, the number of instruments, whether a fixed schema of instruments is required, support for drum tracks, and support for polyphonic tracks. Note that we define a track to be a distinct set of musical material that is played by a single instrument, which may be monophonic or polyphonic (i.e. contain multiple notes that sound simultaneously). Clearly, reducing the restrictions on input material increases the usability of a system, as it can accommodate a greater number of musical styles and user workflows.

As shown in Table 7.1, most systems either support a single track or require a fixed schema of instruments. For example, MusicVAE[18] has a fixed schema of instruments, as it is trained to generate bass, melody and drum trios. One exception is MuseNet[16], which supports up to 10 tracks and any subset of the 10 available instruments. However, there are significant differences between MuseNet and MMM. MuseNet uses separate `NOTE_ON` and `NOTE_OFF` tokens for each pitch on each track, placing inherent limitations on the number of tracks that can be represented, as the token vocabulary size cannot grow unbounded. Considering that MuseNet is currently the largest (in terms of number of weights) music generation model, the number of tracks is unlikely to be increased without altering the representation. Instead, we decouple track information from `NOTE_ON` and `NOTE_OFF` tokens, allowing the use of the same `NOTE_ON` and `NOTE_OFF` tokens in each track. Although this is a relatively small change, it enables us to accommodate all 128 general MIDI instruments. Furthermore, there is no inherent limit on the number of tracks, as long as the entire n-bar multi-track sequence can be encoded using less than 2048 tokens. Practically, this means more than 10 tracks can be generated at once depending on their content. Both MuseNet and MMM do not require a fixed instrument schema, however, MuseNet treats instruments selections as a suggestion, while MMM guarantees a particular instrument will be used.

7.2.2 Generation Methods

We consider four different generation methods: unconditional generation, continuation, infilling and attribute control. Unconditioned generation is analogous to generating music from scratch. Besides changing the data that the model is trained on, the user has limited control over the output of the model. Continuation involves conditioning the model with musical material that precedes (temporally) the music that is to be generated. Since both unconditioned generation and continuation come for free with any auto-regressive model trained on a temporally ordered sequence of musical events, most systems are capable of generating musical material in this manner. Infilling, occasionally referred to as inpainting, conditions generation on a subset of musical material, asking the model to fill in the blanks, so to speak. Note that infilling can occur at different levels (i.e. note-level, bar-level, track-level). Track-level infilling is the most coarse, and allows a set of n-tracks to be generated that are conditioned on a set of k-tracks. Bar-level and Note-level infilling allow for n-bars (resp. notes) selected across one or more tracks to be re-generated, conditioned on the remaining bars (resp. notes) in all other tracks. Attribute-control involves conditioning generation on high-level attributes

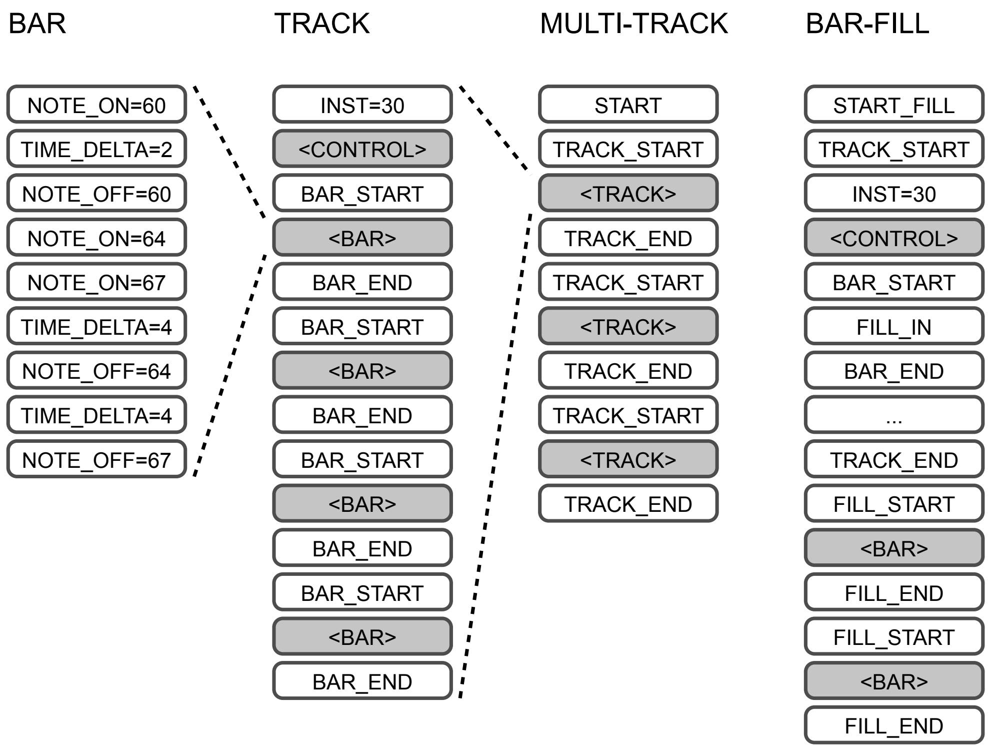


Figure 7.1: The MultiTrack and BarFill representations are shown. The `<bar>` tokens correspond to complete bars, the `<track>` tokens correspond to complete tracks, and the `<CONTROL>` tokens refer to attribute control tokens.

such as style, tempo or density. For example, music generated by MuseNet [16] can be conditioned on a musical style. As shown in Table 7.1, with the exception of SketchNet[4], it is rare to find systems that support both infilling and attribute control. The current version of the MMM supports both track-level and bar-level infilling, and allows for attribute control over instrument, note density, polyphony and note duration. However, we are also actively experimenting with additional attribute controls. Collectively, these improvements afford the end-user a high-degree of control over the generated material, which has previously been proposed as a critical area of research [2].

7.3 Proposed Representation

To provide a comprehensive overview of the proposed representation, we first describe how a single bar of musical material is represented. Based on representations explored in previous studies [13, 10], we represent musical material using 128 `NOTE_ON` tokens, 128 `NOTE_OFF` tokens, and 48 `TIME_SHIFT` tokens. Since musical events are quantized using 12 subdivisions per beat, 48 `TIME_SHIFT` tokens allow for the representation of any rhythmic unit from sixteenth note triplets to a full 4-beat bar of silence. Each bar begins with a `BAR_START` token, and ends with a `BAR_END` token. Tracks are simply a sequence of bars delimited by `TRACK_START` and `TRACK_END` tokens. At the start of each track, immediately following the `TRACK_START` token, an `INSTRUMENT` token is used to specify the MIDI program which is to be used to play the notes on this particular track. Since there are 128 possible MIDI programs, we have 128 distinct `INSTRUMENT` tokens. Tokens which condition generation of each track on various musical attributes follow the `INSTRUMENT` token, which will be discussed in Section 7.4. A multi-track piece is simply a sequence of tracks, however, note that all tracks sound simultaneously rather than being played one after the other. A piece begins with the `START` token. This process of nesting bars within a track and tracks within a piece is illustrated in Figure 1a. Notably, we do not use an `END` token, as we can simply sample until we reach the n^{th} `TRACK_END` token if we wish to generate n tracks. We refer to this representation as the MultiTrack representation.

Using the MultiTrack representation, the model learns to condition the generation of each track on the tracks which precede it. At generation time, this allows for a subset of the musical material to be fixed while generating additional tracks. However, while the MultiTrack representation offers control at the track level, it does not allow for control at the bar level, except in cases where the model is asked to complete the remaining bars of a track. Without some changes, it is not possible to generate the second bar in a track conditioned on the first, third, and fourth bars. In order to accommodate this scenario, we must guarantee that the bars on which we want to condition precede the bars we wish to predict in the sequence of tokens that is passed to the model. To do this, we remove all the bars which are to be predicted from the piece, and replace each bar with a `FILL_PLACEHOLDER` token. Then, at the end of the piece (i.e. immediately after the last `TRACK_END` token), we insert each bar, delimiting each bar with `FILL_START` and `FILL_END` tokens instead of `BAR_START` and `BAR_END` tokens. Note that these bars must appear in the same order as the they appeared in the original MultiTrack representation, shown in Figure 1b. We refer to this representation as the BarFill representation. Note that the MultiTrack representation is simply a special case of the BarFill representation, where no bars are selected for inpainting. To differentiate the BarFill representation from the MultiTrack representation, a `START_FILL` token is used instead of a `START` token.

7.4 Attribute Control

The premise behind attribute controls is quite simple. Given a musical excerpt x, and a measurable musical attribute a for which we can compute a categorical or ordinal value from x (i.e. $a(x)$), the model will learn the conditional relationship between tokens representing $a(x)$ and the musical material on a track, provided these tokens precede the musical material. Practically, this is accomplished by inserting one or more tokens which specify the level of a particular musical attribute $a(x)$ immediately after the `INSTRUMENT` token (see Figure 7.1), and before the tokens which specify the musical material. As a result, our approach is most certainly not limited to the specific musical attributes we discuss below, and can be applied to control any musical feature that can be measured. We employ three approaches to control musical attributes of the generated material: categorical controls, which condition generation on one of n different categories; value controls, which condition generation on one of n different ordinal values; and range controls, which condition the system to generate music wherein a particular musical attribute has values that fall within a specified range. Note that value controls could be considered a special case of range controls, where the lower bound and upper bound on the range are equivalent.

Instrument control is an example of a categorical control, as one of 128 different instrument types can be selected. We use a value control for note density, however, the density categories are determined relative to the instrument type, as average note density varies significantly between instruments. For each of the 128 general MIDI instruments, we calculate the number of note onsets for each bar in the dataset. We divide the distribution for each instrument σ into 10 regions with the range $[P_{10i}(\sigma), P_{10(i+1)}(\sigma))$ for $0 \leq i < 10$, where $P_n(\sigma)$ denotes the n^{th} percentile of the distribution σ. Each region corresponds to a different note density level.

We choose to apply range controls to note duration and polyphony. Each note duration (d) is quantized as follows $\lfloor \log_2(d) \rfloor$. The quantization process groups note durations into 5 different bins $[\frac{1}{32}, \frac{1}{16}), [\frac{1}{16}, \frac{1}{8}), [\frac{1}{8}, \frac{1}{4}), [\frac{1}{4}, \frac{1}{2})$ and $[\frac{1}{2}, \frac{1}{1})$, which we will refer to as note duration levels. Then the 15^{th} and 85^{th} percentiles of a distribution containing all note duration levels within a track is used to condition generation. Polyphony level follows a similar approach. The number of notes simultaneously sounding (i.e. polyphony level) at each timestep is calculated (a timestep is one 16^{th} note triplet). Then we use the 15^{th} and 85^{th} percentiles of a distribution containing all polyphony levels within a track. For both these controls, we use two tokens, one to specify the lower bound and another for the upper bound. Admittedly, this is fuzzy range control, as strict range control would typically use the smallest and largest values in the distribution (0^{th} and 100^{th} percentiles respectively). We elected to use the 15^{th} and 85^{th} percentiles in order to mitigate the effect of outliers within the distribution, decreasing the probability of exposing the model to ranges in which values are heavily skewed to one side of the range.

7.5 Training MMM

We use the MetaMIDI Dataset (Chap. 3) to train our model. We train a GPT2 [17] model using the HuggingFace Transformers library [23] with 8 attention heads, 6 layers, an embedding size of 512, and an attention window of 2048 tokens. Each time we select a n-bar segment during training, we randomly order the tracks so that the model learns each possible conditional ordering between different types of tracks. We also select a random subset of bars for inpainting, masking up to 0.75$\%$ of the bars in a single excerpt. Training typically takes 2-3 days using 4 V100 GPUs.

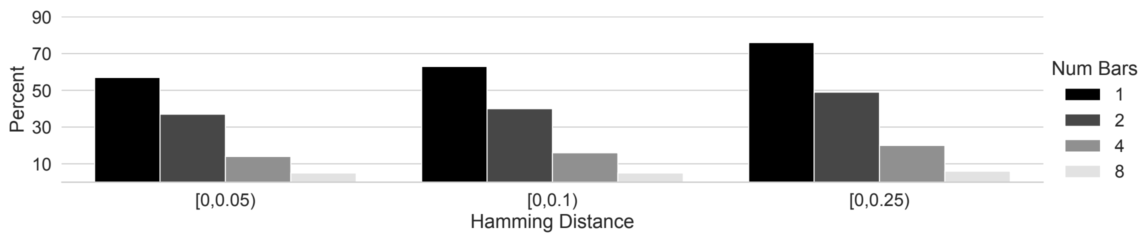


Figure 7.2: The percentage of generated excerpts (g_i) for which the Hamming distance between any excerpt from the training dataset and g_i is on the range [a,b). A Hamming distance of 0 indicates two excerpts are identical, while 1 indicates they are very different.

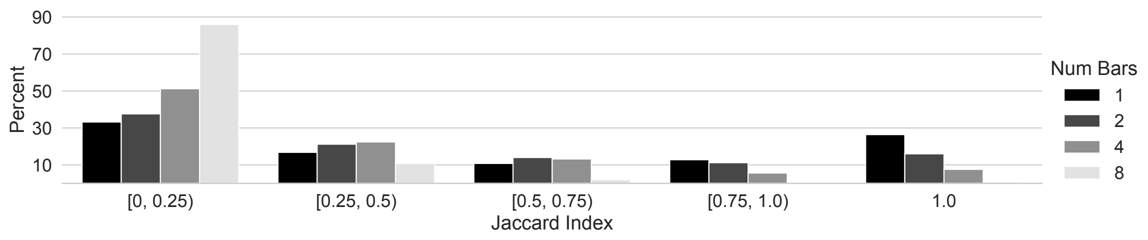


Figure 7.3: The percentage of generated excerpts with a Jaccard Index $\mathcal{J}(o_i, g_i)$ on the range [a,b). A Jaccard Index of 1.0 indicates two excerpts are identical, while 0 indicates they are very different.

7.6 Evaluation and Applications

MMM has seen real-world usage in several contexts, which directly supports our assertion that MMM is a powerful tool for computationally assisted composition. There is ongoing work involving the integration of MMM into synthesizers, audio engines for game design, and a digital audio workstation. MMM has been integrated into a web application, and an Ableton plugin has been developed. MMM has been used to compose music including an album realized with a prototype software integrating MMM into Teenage Engineering synthesizers. More details on the real-world usage of MMM can be found elsewhere[1]. We also make available examples[2] generated by the system, as well as an interactive demo in Google collab[3] that allows one to generate music with MMM.

Bougueng Tchemeube et al. [1] conducted a user-study to evaluate an integration of MMM into Cubase, a popular digital audio workstation. The study measured usability, user experience and technology acceptance for two groups of experienced composers; hobbyists and professionals. Participants conducted three tasks: producing an arrangement by adding generated tracks to an existing track, producing a variation of a pre-existing 16-bar multi-track composition, and creating an original 16-bar composition. One of the strongest findings in the study was that participants

[1]https://metacreation.net/mmm-examples/

[2]https://jeffreyjohnens.github.io/MMM/

[3]https://colab.research.google.com/drive/10ZAdEwHDbL1lVcUGeCdj9FxXnQSNFSH4?usp=sharing

enjoyed using the system, relaying experiences of positive surprises that were generated. However, the study also highlighted room for improvement when it comes to steering MMM's generations in a particular direction. Since Bougueng Tchemeube et al. has already conducted a comprehensive user-study, we do not conduct a listening study for MMM in this chapter. Instead, our experiments are designed to address other aspects which impact the usability of the system in a real-world setting.

Our evaluation of MMM gauges performance of the system by addressing the following research questions:

1. Originality: Does MMM generate original variations or simply duplicate material from the dataset?
2. Stylisitc Similarity: Does MMM generate musical material that is stylistically similar to the dataset?
3. Attribute Controls: How effective are density level, polyphony range and note duration range controls?

7.6.1 Evaluating the Originality of Generated Material

Intra-Dataset Originality

It is increasingly prescient to quantify the frequency with which a generative system is producing musical material that is nearly identical to the training dataset, given potential legal issues that may arise when these systems are deployed into the real world, and the difficulty of guaranteeing that a generative system does not engage in this type of behaviour [14].

Alternate approaches for measuring originality have been proposed. Collins and Laney proposed the Cardinality Score [6], which is based on a transposition invariant measure of similarity between two musical excerpts, with notes represented as a series of onset-time and pitch tuples. One disadvantage of this approach is that it if two sets of notes differ from one another by a small non-constant amount, the similarity will be low, despite being very similar perceptually. Fortunately, this is not an issue for our approach as each note is represented using multiple cells in a piano roll matrix, rather than representing notes using onset-time and pitch tuples. Another more promising approach, is the Originality Report [24], which uses symbolic fingerprinting to measure the similarity between musical excerpts. However, we elected not to use this approach for several reasons. First, the run-time of the Originality Report would be quite substantial when working with a large dataset like the MMD. According to Yin et al., it takes 600 seconds to compute a single query against the Maestro dataset, which is comprised of 962 MIDI files. Using the MMD, which contains 436,631 MIDI files, a single query would take approximately 37 hours. Note that our estimate accounts for the fact that the average MIDI file in the Maestro dataset contain 2 times more notes than those in the MMD. Moreover, since our experiment involves computing hundreds of queries, this approach quickly becomes intractable on a dataset as large as the MMD. Second, the Originality Report is not designed to accommodate multi-track music. The symbolic fingerprinting method hashes sets

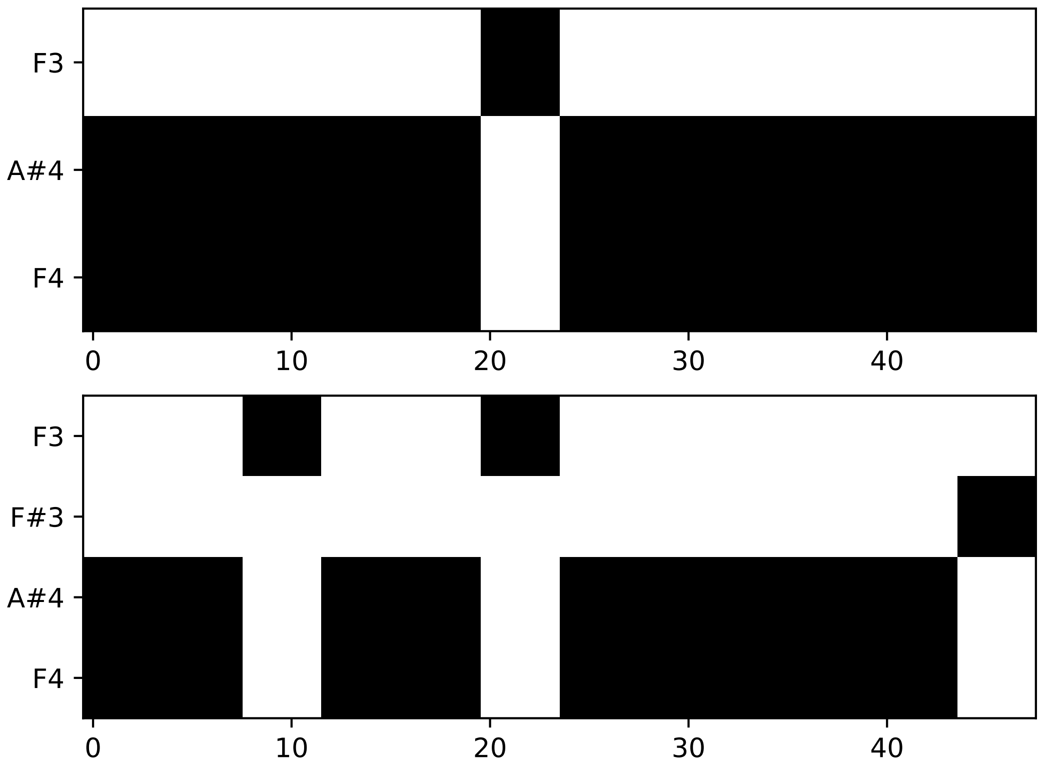


Figure 7.4: Two bars with a Jaccard Index of 0.76 and a normalized Hamming Distance of 0.25.

of three note onset and pitch tuples. To avoid selecting sets that span multiple parts or voices, the authors filter sets based on pitch intervals and time intervals. This works well for string quartets and piano compositions where individuals parts or voices rarely overlap, but is ill-suited to multi-track music where note ranges on different tracks frequently overlap. Although there are surely ways to address this issue, it is likely that these solutions would further impact the run-time of the algorithm. Consequently, we elect to develop our own approach that is suitable for large multi-track MIDI datasets.

In Section 7.6.1, musical material is represented using a piano roll, which is a $T \times 128$ boolean matrix specifying when particular pitches are sounding, where T is the number of time-steps. Note that when we calculate Hamming distance between two piano rolls, we normalize the distance by the number of non-zero cells in the query piano roll. Therefore, the distance between maximally different piano rolls is 1. In Figure 7.4, we provide an example of two bars and the normalized Hamming Distance between them.

Since the dataset contains hundreds of thousands of unique MIDI files, we are faced with a time complexity issue, and must employ some heuristics to speed this process up. First, rather than searching nearly identical n-bar piano rolls, we search using single-bar piano rolls, and aggregate the results of n search processes. Note that this means that if $n-1$ of the bars have a match in the dataset, but one of the bars does not, the n-bar excerpt will not be considered to have a match in the dataset. However, since we are interested in identifying nearly identical matches, this is unlikely to cause much of an issue. To filter out highly dissimilar candidate matches efficiently, we compute the Hamming distance between compressed piano rolls first. Given a 48×128 piano roll x that represents a single 4/4 bar of musical material, we discard notes outside the range $[21, 109)$ and take the maximum value over each consecutive set of 6 time-steps (equivalent to one 1/8 note) on the first axis, producing a 8×88 matrix x. We calculate the Hamming distance between the compressed piano rolls, discarding any candidate matches which have a distance greater than 0.25, and then compute Hamming distance on the full sized piano rolls for the remaining candidate matches. Even with these optimizations, the search is executed in parallel a 32 core machine and takes an average of 83 seconds to complete a search for a single 4 bar excerpt. In the worst case, it can take up to an hour for a single query. Although we would have preferred to use Jaccard index rather than Hamming distance, as we do in Section 7.6.1, the nature of the heuristics employed prohibited this option.

We compute 100 trials where we randomly select a 4 track 8 bar musical segment from the test split of the dataset, blank out n consecutive bars on a single track and generate (i.e. infill) a new set of bars (g_i). Given a hamming distance threshold we determine if g_i is nearly identical to any n-bar excerpt in the training split of dataset, using the method described above. In Figure 7.2 we present the percentage of trials for which the Hamming distance between any excerpt in the training dataset and g_i is on the specified range. Unsurprisingly, as the number of bars increases, the percentage of instances where MMM duplicates the training data decrease significantly. This correlation was expected as shorter generations are more constrained by the surrounding musical content.

Infilling Originality

We also consider the case when the original musical material is simply duplicated when infilling, resulting in no change to the musical material, and inevitable frustration on the part of the user. To measure the frequency with which this occurs, we randomly select a 4 track 8 bar musical segment from the test split of the dataset, blank out n consecutive bars on a single track (o_i), and generate a new set of n bars (g_i) to replace (o_i). Then we measure the Jaccard index between piano roll representations of o_i and g_i. We repeat this process 250 times for each number of bars $(n = 1, 2, 4, 8)$ and report the results in Figure 7.3. On a whole, as the number of bars increases, the frequency with which the original material is duplicated decreases. Taken collectively, the results in Section 7.6.1 and 7.6.1 seem to indicate that the MMM can reliably produce original variations when the generating 4 or more bars.

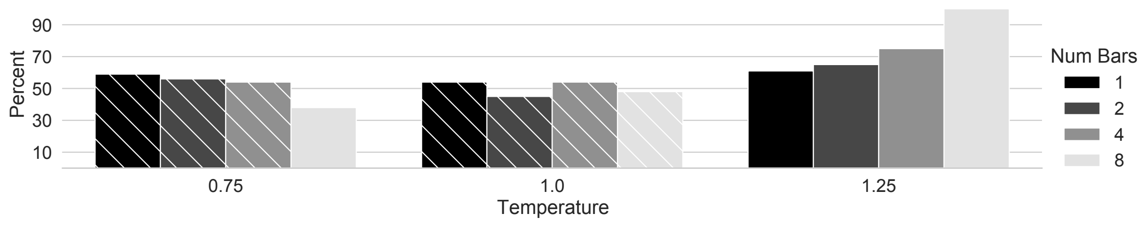


Figure 7.5: The percentage of trials where $\mathcal{S}^{\mathcal{C}^\star_{50}}_{\hat{\mathcal{O}}^\star_{25},\hat{\mathcal{G}}^\star_{25}}(\hat{\mathcal{O}}^\star_{25},\mathcal{C}^\star_{50}) \leq \mathcal{S}^{\mathcal{C}^\star_{50}}_{\hat{\mathcal{O}}^\star_{25},\hat{\mathcal{G}}^\star_{25}}(\hat{\mathcal{G}}^\star_{25},\mathcal{C}^\star_{50})$. Hatching indicates that the binomial test was insignificant, indicating that $\hat{\mathcal{O}}^\star$ is not more similar to $\mathcal{C}$ than $\hat{\mathcal{G}}^\star$.

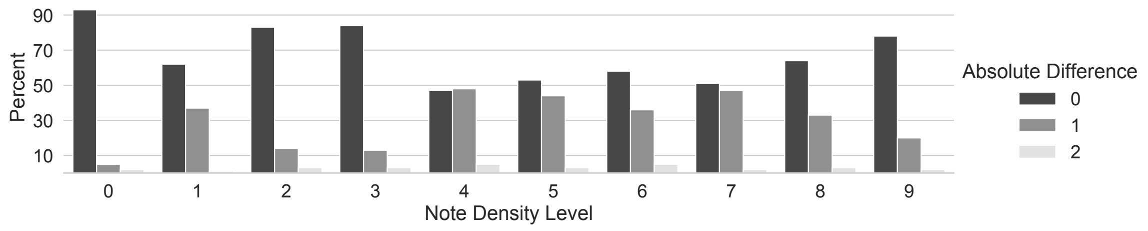


Figure 7.6: The percentage of trials for each absolute difference between anticipated and actual note density level.

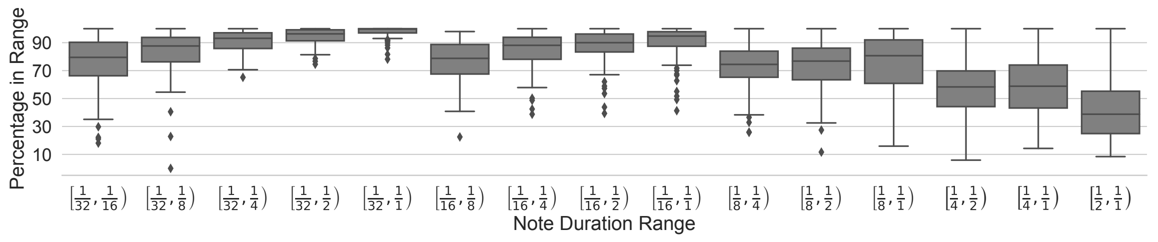


Figure 7.7: The percentage of note durations within the range shown for 100 trials.

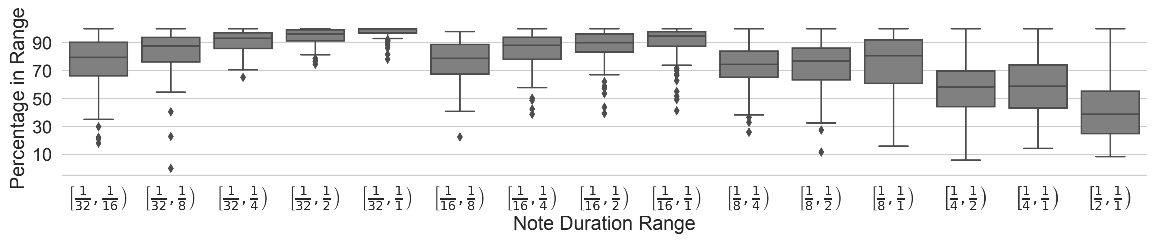


Figure 7.8: The percentage of polyphony levels within the range shown for 100 trials.

7.6.2 Quantifying Stylistic Similarity

It is also important that the variations generated by the system are stylistically similar to the dataset. We use StyleRank (Chap. 5) to measure the stylistic similarity of generated material. StyleRank is designed to measure the similarity of two or more groups of musical excerpts $(\mathcal{G}_1, ..., \mathcal{G}_k)$ relative to a style delineated by a collection of ground truth musical excerpts $(\mathcal{C})$. Each musical excerpt is represented using a set of features, described in detail in the original paper, and a Random Forest classifier is trained to discriminate between $\mathcal{G}_1, ..., \mathcal{G}_k$ and $\mathcal{C}$. Using an embedding space constructed from the trained Random Forest classifier, the average similarity between $\mathcal{G}_i$ and $\mathcal{C}$ can be computed for each i. In what follows, let $\mathcal{S}^{\mathcal{C}}_{\mathcal{G}_1,...,\mathcal{G}_k}(a,b)$ denote the median similarity between a and b, calculated using a StyleRank instance trained on $\mathcal{G}_1, ..., \mathcal{G}_k$ and $\mathcal{C}$.

For this experiment, we use the same musical excerpts from Section 7.6.1 ($\mathcal{O} = \{o_1, ..., o_{250}\}, \mathcal{G} = \{g_1, ..., g_{250}\}$), however, we remove each pair (o_i, g_i) where $\mathcal{J}(o_i, g_i) \geq 0.75$, producing $\hat{\mathcal{O}}$ and $\hat{\mathcal{G}}$. This ensures that we do not bias our measurements by including generated material that is nearly identical to the original preexisting material (o_i) from the dataset. We also assemble a set of 1000 n-bar segments ($\mathcal{C}$) from the dataset. For each trial, we compute $\mathcal{S}^{\mathcal{C}^\star_{50}}_{\hat{\mathcal{O}}^\star_{25},\hat{\mathcal{G}}^\star_{25}}(\hat{\mathcal{O}}^\star_{25}, \mathcal{C}^\star_{50}) \leq \mathcal{S}^{\mathcal{C}^\star_{50}}_{\hat{\mathcal{O}}^\star_{25},\hat{\mathcal{G}}^\star_{25}}(\hat{\mathcal{G}}^\star_{25}, \mathcal{C}^\star_{50})$, where $X^\star_n$ denotes a subset of X containing n elements, which are selected randomly for each trial. We collect the results for 100 trials, and compute a binomial test. An insignificant result indicates that $\hat{\mathcal{O}}$ and $\hat{\mathcal{G}}$ are equally similar to $\mathcal{C}$. We report the results of this test using different number of bars and temperature in Figure 7.5.

The results indicate that when generating with a temperature of 1.0, infilled generations are roughly equivalent to the original preexisting material in terms of musical style (as quantified by StyleRank). Our results also show that when temperature is greater than 1.0, the generated material is more frequently considered less similar to the dataset ($\mathcal{C}$) than $\hat{\mathcal{O}}$, an effect that increases along with the number of generated bars. This demonstrates that our measurement instrument is capable of detecting small differences in musical style, which are the byproduct of slightly changing the temperature, increasing the entropy of the probabilities output by the model.

7.6.3 Evaluating the Effectiveness of Attribute Controls

MMM allows the user to condition generation on various attributes such as note density, polyphony level, and note duration. To evaluate how effective these control mechanisms are, we conduct 100 trials where we generate 8-bar segments using a particular attribute control method, and measure the difference between the anticipated outcome and the actual outcome. For note density control, we measure the absolute difference between the density level the generation was conditioned on and the density level of the generated material. For polyphony level and note duration, we compute the distribution of values (either polyphony level or note duration level) from the generated material, and count the percentage of values that fall within the specified range. If attribute control is successful, we would expect at least 70% of the values to be within this range, as we used the 15^{th} and 85^{th} percentiles while training.

The results for note density, shown in Figure 7.6, demonstrate that the majority of times the absolute difference between the anticipated and actual density level is most often 0, and rarely exceeds 1. This seems to indicate that this control method is quite effective. The results for Note Duration, shown in Figure 7.7 demonstrate that this attribute control method is quite effective, as the median outcome (in terms of percentage of note durations within the specified range) is at or above 70% in all cases except for $[\frac{1}{4}, \frac{1}{2})$, $[\frac{1}{4}, \frac{1}{1})$ and $[\frac{1}{2}, \frac{1}{1})$. In contrast the Polyphony Level control is less effective, with the median outcome lying below the 70% threshold in many cases. Calculating polyphony level at a single time-step is inherently more difficult than note duration, as the former requires knowledge of where multiple notes start and end, while the later only requires knowing where one note starts and ends. This difference in difficulty seems to be reflected in the results, as the MMM is better at controlling note duration than polyphony.

7.7 Limitations and Future Work

Although we pre-processed MIDI files from the MMD by discarding those with irregular time signatures, and quantizing note onsets to a 16th-note triplet grid, this process may be insufficient in some cases. Both the Lakh MIDI dataset and the MMD contain MIDI files that have been performed, and thus, do not always align to a fixed temporal grid, even after quantization. Since research [11] has demonstrated that the performance of a model is bounded by the quality of the training data, future work may involve improvements to the data pre-processing pipeline, to ensure MMM is trained on high quality data exclusively. With regards to our experiments that measure the originality of material generated by MMM, our approach only detects highly similar matches that have been transposed on the range [-6,6) semitones. The basis for this decision was that MMM is trained on excerpts from the dataset that have been transposed on the range [-6,6) semitones. Although our tests would demonstrate if excerpts generated by MMM copy directly from the training data, they would not detect transpositions outside this range. Although the Originality Report [24] is transpositionally invariant and does not suffer from this limitation, the run-time of this approach made it intractable. A tractable solution which can be applied in future work, would involve increasing the transposition range for our method.

7.8 Conclusion

In this paper, we introduced MMM, a generative system which employs a novel approach to representing musical material, resulting in increased control over the generated output. We provided experimental evidence demonstrating the effectiveness of the system, and outlined several ongoing real-world applications. One main limitation is that the model can only generate 8 bars at once. However, using an auto-regressive approach longer pieces can be generated by repeatedly conditioning on some portion of the preexisting material. Future work involves optimizing the model for real-time generation, expanding the set of attribute controls, and continued integration of MMM into real-world products and platforms.

Bibliography

[1] Renaud Bougueng Tchemeube, Jeff Ens, and Philippe Pasquier. “Evaluating Human-AI Interaction with MMM-Cubase: A Creative AI System for Music Composition”. In: *Under Review at the 32nd International Joint Conference of Artificial Intelligence*. 2023.

[2] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. *Deep Learning Techniques for Music Generation*. Computational Synthesis and Creative Systems. Springer International Publishing, 2019.

[3] Chin-Jui Chang, Chun-Yi Lee, and Yi-Hsuan Yang. “Variable-Length Music Score Infilling via XLNet and Musically Specialized Positional Encoding”. In: *Proceedings of the 22nd ISMIR*. Ed. by Jin Ha Lee, Alexander Lerch, Zhiyao Duan, Juhan Nam, Preeti Rao, Peter van Kranenburg, and Ajay Srinivasamurthy. 2021, pp. 97–104.

[4] Ke Chen, Cheng-i Wang, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. “Music Sketch-Net: Controllable Music Generation via Factorized Representations of Pitch and Rhythm”. In: *Proceedings of the 21st ISMIR*. Ed. by Julie Cumming, Jin Ha Lee, Brian McFee, Markus Schedl, Johanna Devaney, Cory McKay, Eva Zangerle, and Timothy de Reuse. 2020, pp. 77–84.

[5] Wayne Chi, Prachi Kumar, Suri Yaddanapudi, Rahul Suresh, and Umut Isik. “Generating Music with a Self-Correcting Non-Chronological Autoregressive Model”. In: *Proceedings of the 21st ISMIR*. Ed. by Julie Cumming, Jin Ha Lee, Brian McFee, Markus Schedl, Johanna Devaney, Cory McKay, Eva Zangerle, and Timothy de Reuse. 2020, pp. 893–900.

[6] Tom Collins and Robin Laney. “Computer-generated stylistic compositions with long-term repetitive and phrasal structure”. In: *Journal of Creative Music Systems* 1.2 (2017).

[7] Chris Donahue, Huanru Henry Mao, Yiting Ethan Li, Garrison W Cottrell, and Julian McAuley. “LakhNES: Improving multi-instrumental music generation with cross-domain pre-training”. In: *Proc. of the 20th International Society for Music Information Retrieval Conference*. 2019, pp. 685–692.

[8] Yongjun Hong, Uiwon Hwang, Jaeyoon Yoo, and Sungroh Yoon. “How Generative Adversarial Networks and Their Variants Work: An Overview”. In: *ACM Computing Surveys* 52.1 (2019), 1:43.

[9] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron C. Courville, and Douglas Eck. “Counterpoint by Convolution”. In: *Proceedings of the 18th International Society for Music Information*. 2017, pp. 211–218.

[10] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. “Music Transformer: Generating Music with Long-Term Structure”. In: *7th International Conference on Learning Representations*. 2019.

[11] Abhinav Jain, Hima Patel, Lokesh Nagalapatti, Nitin Gupta, Sameep Mehta, Shanmukha Guttula, Shashank Mujumdar, Shazia Afzal, Ruhi Sharma Mittal, and Vitobha Munigala. “Overview and importance of data quality for machine learning tasks”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 3561–3562.

[12] Gautam Mittal, Jesse H. Engel, Curtis Hawthorne, and Ian Simon. “Symbolic Music Generation with Diffusion Models”. In: *Proceedings of the 22nd ISMIR*. Ed. by Jin Ha Lee, Alexander Lerch, Zhiyao Duan, Juhan Nam, Preeti Rao, Peter van Kranenburg, and Ajay Srinivasamurthy. 2021, pp. 468–475.

[13] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. “This time with feeling: learning expressive musical performance”. In: *Neural Computing and Applications* (2018), pp. 1–13.

[14] Alexandre Papadopoulos, Pierre Roy, and François Pachet. “Avoiding Plagiarism in Markov Sequence Generation”. In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. Ed. by Carla E. Brodley and Peter Stone. AAAI Press, 2014, pp. 2731–2737.

[15] Ashis Pati, Alexander Lerch, and Gaëtan Hadjeres. “Learning to Traverse Latent Spaces for Musical Score Inpainting”. In: *Proc. of the 20th International Society for Music Information Retrieval Conference*. 2019, pp. 343–351.

[16] Christine Payne. “MuseNet”. In: *OpenAI* (Apr. 2019). openai.com/blog/musenet.

[17] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. “Language models are unsupervised multitask learners”. In: *OpenAI Blog* 1.8 (2019), p. 9.

[18] Adam Roberts, Jesse H. Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. “A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music”. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 4361–4370.

[19] Bob L Sturm, Oded Ben-Tal, Úna Monaghan, Nick Collins, Dorien Herremans, Elaine Chew, Gaëtan Hadjeres, Emmanuel Deruty, and François Pachet. “Machine learning research that matters for music creation: A case study”. In: *Journal of New Music Research* 48.1 (2019), pp. 36–55.

[20] Hao Hao Tan and Dorien Herremans. “Music FaderNets: Controllable Music Generation Based On High-Level Features via Low-Level Feature Modelling”. In: *Proceedings of the 21st ISMIR*. Ed. by Julie Cumming, Jin Ha Lee, Brian McFee, Markus Schedl, Johanna Devaney, Cory McKay, Eva Zangerle, and Timothy de Reuse. 2020, pp. 109–116.

[21] Ziyu Wang, Dingsu Wang, Yixiao Zhang, and Gus Xia. “Learning Interpretable Representation for Controllable Polyphonic Music Generation”. In: *Proceedings of the 21st ISMIR*. Ed. by Julie Cumming, Jin Ha Lee, Brian McFee, Markus Schedl, Johanna Devaney, Cory McKay, Eva Zangerle, and Timothy de Reuse. 2020, pp. 662–669.

[22] Ziyu Wang and Gus Xia. “MuseBERT: Pre-training Music Representation for Music Understanding and Controllable Generation”. In: *Proceedings of the 22nd ISMIR*. Ed. by Jin Ha Lee, Alexander Lerch, Zhiyao Duan, Juhan Nam, Preeti Rao, Peter van Kranenburg, and Ajay Srinivasamurthy. 2021, pp. 722–729.

[23] Thomas Wolf et al. “HuggingFace’s Transformers: State-of-the-art Natural Language Processing”. In: *ArXiv* abs/1910.03771 (2019).

[24] Zongyu Yin, Federico Reuben, Susan Stepney, and Tom Collins. “Measuring When a Music Generation Algorithm Copies Too Much: The Originality Report, Cardinality Score, and Symbolic Fingerprinting by Geometric Hashing”. In: *SN Computer Science* 3.5 (2022), p. 340.

Chapter 8

Conclusion

8.1 Summary

This thesis presents contributions related to the evaluation of CSMG systems (Chapters 4, 5, 6), as well as the MetaMIDI Dataset (Chapter 3), and the Multi-Track Music Machine (Chapter 7). In Chapter 1, we specify the structure of the thesis, outlining the motivations and contributions. Chapter 2 provides an overview of related work in the area of CSMG systems, detailing the data, representation, architectures and evaluation methods that have previously been employed. Chapter 3 details the process of assembling the MetaMIDI Dataset, the largest publicly available MIDI dataset to date, comprised for over 440,000 MIDI files and an expanded set of metadata for many of the MIDI files. CAEMSI is introduced in Chapter 4, which is a domain-agnostic analytic evaluation methodology for style imitation, which employs permutation testing to measure the statistical equivalence or difference between two sets of artifacts. StyleRank is described in Chapter 5, a system for ranking symbolic musical excerpts based on their similarity to a style, as delineated by an arbitrary collection of MIDI files. Chapter 6 advances recommendations for listening test design, motivated by a lack of consensus within the research community, as evidenced by inconsistencies across listening study experimental designs. Finally, the Multi-Track Music Machine is proposed in Chapter 7, and StyleRank is employed as part of the evaluation process.

8.2 Applications of the Multi-Track Music Machine

The Multi-Track Music Machine (MMM) has been integrated into several related areas of research including: Calliope [1], an online computer-assisted composition system for music generation; and studies on generative music for game design. In addition, we have several ongoing collaborations with industry partners, who are interested in integrating MMM into their products.

8.2.1 Calliope

The Calliope system [1] is a co-creative interface for multi-track music generation, which uses the Multi-Track Music Machine to generate musical material. It provides many features that are similar to a standard DAW, such as the ability to view and play MIDI files, as well as add and remove tracks. However, it is also designed with the features of MMM in mind, providing a convenient interface for bar and track infilling, and an interface for controlling various hyper-parameters which influence the generation process. Calliope also integrated StyleRank [8], which is used to rank a collection of MIDI files with respect to their musical similarity to a target MIDI file. This can be especially useful when Calliope's batch generation mode is employed to generate many different variations, as they can be ranked based on similarity to the original.

8.2.2 Affect Models for Game Design

The research of Plut et al. [11] used the MMM system to expand a manually composed score into a generative score. To accomplish this, manually composed MIDI files were provided to MMM and bars were progressively infilled to create many variations of the original MIDI file. The resulting generative score was incorporated into a game (Galactic Deffense) that was used by Plut et al. to evaluate the Predictive Gameplay-based Layered Affect Model (PreGLAM) in a real-world setting.

8.2.3 Industry Collaborations

There are ongoing collaborations with industry partners which involve integrating MMM into various products, and exploring issues related to the acceptability and usability of these systems in the real-world. With Steinberg, which develops Cubase, we have developed a streamlined integration of MMM, and are currently running a user-experiment which measures the acceptability and usability of the system within this context. With another industry partner we are exploring the integration of MMM into a portable audio synthesizer. Finally, we are also collaborating with another industry partner who is interested in integrating MMM into software that is used to create adaptive music for video games. An Ableton plugin has also been developed for MMM.

8.2.4 Music Composition

MMM has been also used to compose music including: an album [1] realized with a prototype software integrating MMM into Teenage Engineering synthesizers; and an entry into the AI Song Contest [2].

8.3 Limitations and Future Work

In this section, we discuss limitations and opportunities for future work related to the research presented in this thesis.

[1]https://open.spotify.com/artist/3AcNnEmImsLGiOQbv9r3Ha

[2]https://www.aisongcontest.com/participants-2022/monobor-x-mashmachine

8.3.1 MetaMIDI Dataset

The MIDI files contained in the MetaMIDI Dataset (MMD) and the Lakh MIDI Dataset (LMD) are heterogeneous, in stark contrast to smaller homogeneous datasets like the Maestro Dataset [6]. Although some may view the heterogeneity of the MMD and LMD as a cause for concern, in our view, this is a positive feature of both datasets. First, one of the primary contributions of these datasets is that they aggregate and de-duplicate a large collection of MIDI files, which were previously dispersed across hundreds of different websites. Second, the heterogeneous nature of the data means that these datasets can support the development of a variety of generative systems. Moreover, once the data has been aggregated, it is relatively easy for researchers to define a pre-processing pipeline that suits the particular needs of the research project. For example, although non-quantized MIDI files may be of little use for developing a system that generates musical scores, they are essential for developing systems that generate music that sounds like a human performance. Rather than implicitly imposing a specific research agenda on the users of the dataset by curating or pre-processing the MIDI files, we elect to provide the MIDI files as is.

8.3.2 CSMG Evaluation Methods

A substantive portion of the thesis was devoted to the development of analytic evaluation methods for CSMG systems [7, 8]. Although these methodologies were specifically designed to address the limitations of the prototypical listening test, they are not without their own limitations.

CAEMSI is based on the Frequentist Hypothesis Testing (FHT) paradigm, which introduces some limitations when testing for equivalence. When conducting an equivalence test using CAEMSI, an equivalence interval must be defined, that specifies the range on which the two distributions will be considered equivalent. In general, choosing an appropriate equivalence interval is domain dependant. To complicate matters further, CAEMSI is technically distance metric agnostic. As a result, when determining a suitable equivalence interval, one must also account for the distance metric that is being employed. Consequently, alternate approaches that are based on Bayes Factor Analysis (BFA) may be more appropriate solution. For example, Yin et al. [14] have used non-parametric equivalence tests based on the BFA paradigm to compare several generative music systems. However, we must note that the non-parametric tests [4] used by Yin et al. were unavailable at the time our research was being conducted, as they were published several years later.

In the chapter that introduces CAEMSI, we use Normalized Compression Distance [10] (NCD), which is a highly generalized distance metric that is domain agnostic. Unsurprisingly, when using such a metric, there are limitations when applied to specific domains. For example, in a musical context it is often desirable to account for transposition or time shift when measuring the distance between two excepts, however, NCD is unable to account for these types of transformations. As a result, NCD(X,Y) would not be at a minimum when Y is a transposition of X, or when Y is a time-stretched version of X. Since NCD is based on information theory and utilizes compression to measure the distance between two inputs, it is unlikely that it corresponds to human perception uniformly. With respect to this limitation, it is important to reiterate that CAEMSI can accommodate any distance metric, so it is possible to use a distance metric that is more suited to a particular domain. Importantly, the results presented in Chapter 4 demonstrate that NCD can discriminate between the work of two different composers, despite the domain specific limitations discussed above. However, since this thesis focuses on the music domain, NCD's limitations with respect to common musical transformations directly motivated the development of StyleRank, a domain specific solution.

StyleRank represents musical material using a set of features, however, these features may be inadequate for some types of music, as the experiments which validated StyleRank only used classical music. Furthermore, we feel it is necessary to moderate our claim in Section 5.9 that "[StyleRank] is highly correlated with human perception of stylistic similarity", as the experimental results only demonstrate a correlation with human perception of stylistic similarity for J.S. Bach chorales, which certainly does not guarantee generalization across all musical styles. Although StyleRank appears to work effectively with the other genres of music present in the MetaMIDI dataset, an expanded evaluation of StyleRank was beyond the scope of this thesis.

To validate both CAEMSI and StyleRank, we employed a test where we measured their ability to discriminate between the work of two different composers. Admittedly, when evaluating a generative music system, we are testing for a much smaller difference, as we aim to measure the difference between human-composed music and generated music in the same style as the human composer. As with human participants, there are inherent limitations to the differences that can be perceived when comparing musical stimulus, and this limitation may have some impact on the results. However, the results presented in section 7.6.2 do indicate that StyleRank is capable of detecting the difference between generated and human-composed music when the temperature is slightly increased, which directly supports the claim that these evaluation methods are suitable for the task. Future work in this area could involve a larger studies comparing the results of human-listening experiments with these analytic methods across a wide variety of musical genres.

When discussing the motivating factors for CAEMSI and StyleRank, we emphasize the limitations of listening studies with respect to domain knowledge, human bias against generative systems, variability and scalability. However, it is important to acknowledge that human assessments are the gold standard when it comes to evaluating generative music systems. As a result, any analytic metric or methodology must be validated against human perception. Undoubtedly, this is a fundamental limitation of any analytic evaluation method, as we first must demonstrate that it corresponds to human perception, before it can be applied.

8.3.3 Optimization of the Multi-Track Music Machine

Although we are satisfied by the generative capabilities of MMM, the computational demands of the system impose some limitations on where it can be deployed. When processing larger sections of musical material, MMM can require up to 32GB of memory, which currently prohibits deployment on many devices. Furthermore, the time which is required to process a generative request currently prohibits real-time applications.

There are several promising areas of future work, which may assist in mitigating issues related to memory consumption and running time. First, it would be worthwhile to experiment with replacing the GPT-2 [12] architecture, which has an attention mechanism [13] with quadratic complexity, with an architecture that reduces the computational complexity of the attention mechanism [9, 3, 2]. Second, the knowledge distillation [5] technique could be used to train a smaller network to replicate the probability distributions output by the original network using inputs from the training dataset. Another possibility, is to employ compressed vector-based representations of musical material, allowing a set of bars to be represented using a set of fixed-size vectors. These vectors could be used to represent portions of the musical context instead of the lengthy discrete token sequences that are currently used. This would reduce memory consumption and running time without major changes to the architecture. Since the development of attention-based architectures and knowledge-distillation training curriculum are active areas of research, there are a myriad of possibilities to be explored, which was simply beyond the scope of this thesis, and have thus been left to future work.

8.3.4 Improving Attribute Control in the Multi-Track Music Machine

The current attribute controls for note duration and polyphony level can be considered "fuzzy" controls. Since minimum and maximum are not robust estimators of the bounds of a distribution, as these values can be skewed by a single outlier, we elected to use the 15% and 85% percentiles to determine the bounds of note duration and polyphony level distributions. Collecting informal feedback from users of MMM made it clear that the trade-off for statistical validity, was users ability to understand and interpret these controls. Unfortunately, in some cases, directly contributed to a lack of trust in the system, which is undoubtedly something important to address in future iterations of the system.

Future work in this area may involve using hard controls that prohibit MMM from generating note durations or polyphony levels outside of the specified range. Since the framework for integrating attribute controls is quite general, as we can train a model to respect any control specified by a programmatic function that can assign a discrete valued output to each item in the training dataset, additional attribute controls may be developed in the future. These may include controls for: genre, silence amount, pitch range, musical key, melodic complexity, rhythmic complexity, valence, arousal and tension. It is also worth exploring controls using vectors of continuous values, as this will accommodate control over domains that are not best represented using discrete values.

8.3.5 Expanding the Rhythmic Capabilities of the Multi-Track Music Machine

When assembling the MIDI files from which the MetaMIDI dataset is comprised, a specific effort was not made to include MIDI files exclusively containing rhythmic musical material. Since MMM was trained using the MetaMIDI dataset, it may be worth aggregating rhythmic MIDI files, and including these files in the training dataset to improve the rhythmic generation capabilities of MMM.

8.3.6 Performative Interpretation with the Multi-Track Music Machine

MMM is trained using MIDI files that are quantized to the nearest sixteenth note triplet, and thus generates quantized musical material. Future work will likely involve exploring various approaches to integrating performed interpretation of note timing, as well as velocity.

Bibliography

[1] Renaud Bougueng Tchemeube, Jeff Ens, and Philippe Pasquier. “Calliope: An Online Generative Music System for Symbolic Multi-TrackComposition”. In: *Proceedings of the International Conference on Computational Creativity* (2022).

[2] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. “Generating long sequences with sparse transformers”. In: *arXiv preprint arXiv:1904.10509* (2019).

[3] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. “Rethinking attention with performers”. In: *arXiv preprint arXiv:2009.14794* (2020).

[4] Johnny van Doorn, Alexander Ly, Maarten Marsman, and E-J Wagenmakers. “Bayesian rank-based hypothesis testing for the rank sum test, the signed rank test, and Spearman’s ρ”. In: *Journal of Applied Statistics* 47.16 (2020), pp. 2984–3006.

[5] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. “Knowledge distillation: A survey”. In: *International Journal of Computer Vision* 129.6 (2021), pp. 1789–1819.

[6] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse H. Engel, and Douglas Eck. “Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset”. In: *7th International Conference on Learning Representations*. 2019.

[7] **Jeff Ens** and Philippe Pasquier. “CAEMSI: A Cross-Domain Analytic Evaluation Methodology for Style Imitation.” In: *Proceedings of the International Conference on Computational Creativity*. 2018, pp. 64–71.

[8] **Jeff Ens** and Philippe Pasquier. “Quantifying Musical Style: Ranking Symbolic Music based on Similarity to a Style”. In: *Proc. of the International Symposium on Music Information Retrieval*. 2019, pp. 870–877.

[9] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. “Reformer: The efficient transformer”. In: *arXiv preprint arXiv:2001.04451* (2020).

[10] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul MB Vitányi. “The similarity metric”. In: *IEEE transactions on Information Theory* 50.12 (2004), pp. 3250–3264.

[11] Cale Plut, Philippe Pasquier, Jeff Ens, and Renaud Bougueng. “Preglam: A Predictive, Gameplay-Based Layered Affect Model”. In: *Gameplay-Based Layered Affect Model* ().

[12] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language models are unsupervised multitask learners". In: *OpenAI Blog* 1.8 (2019), p. 9.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[14] Zongyu Yin. "New evaluation methods for automatic music generation". PhD thesis. University of York, 2022.

Appendix A

Cumulative Dissertation information

SIAT Guidelines

WRITING A CUMULATIVE THESIS

Produced by the SIAT Graduate Program Committee, 6 November 2013, amended May 2021

This document is adjunct to SIAT's calendar entry and to SFU's Graduate General Regulations. It describes SIAT's normal practice with respect to the topic it addresses.

It has been approved by SIAT's Graduate Caucus.

The PhD and Masters thesis may have the form of a monograph (i.e., the classic thesis format of one single document), or of a compilation with a number of scholarly peer-reviewed articles ("cumulative thesis"). Students should always consult early on with their senior supervisor and committee, who will discuss with them and decide what form of thesis is the most suitable for a given case.

Please note that, the guidelines suggested here are to outline possible criteria and courses of actions when preparing a cumulative thesis. The students' supervisory committee might have different criteria or requirements and will decide on what contributions constitute a thesis.

1. In the case of a cumulative thesis, the selected scholarly articles are to be connected by an initial introduction chapter (explaining the subject and scope of the work and how the different articles contribute) and a summative final discussion chapter (that includes the overall contributions, main conclusions, and an outlook). This serves to interrelate the contributions as well as discuss and draw conclusions from the entire work. The publications need to be integrated as chapters into the theme of the thesis and must deal with the overall topic of the thesis. The thesis should have continuous pagination and an aggregated bibliography. The individual papers may still have their own bibliography in addition.

2 a) Phd Thesis: The PhD thesis should have a content and depth corresponding to a classic (monograph-style) thesis. E.g., this might be achieved by about 3-6 peer-reviewed conference papers, journal articles, or other written scholarly contributions of high value where the supervisory committee assesses the quality as being appropriate. The articles should maintain such a level that they could be accepted for publication in an international scholarly journal with a rigorous referee procedure. At least two of these articles should already have been accepted for publication or be published. All may already be published. For published articles, the comprehensive bibliographic reference should be stated. For accepted or submitted manuscripts, the venue and date of acceptance/submission should be included.

2 b) Masters thesis: The Masters thesis should have a content and depth corresponding to a classic (monograph-style) thesis. E.g., this might be achieved by at least 2 peer-reviewed conference papers, journal articles, or other scholarly written contributions of high value where the supervisory committee assesses the quality as being appropriate. The articles should maintain such a level that they could be accepted for publication in an international scholarly journal with a rigorous referee procedure. All of these articles need to be submissible (as judged by the supervisory committee), submitted, accepted, or published. For published articles, the comprehensive bibliographic reference should be stated. For accepted or submitted manuscripts, the venue and date of acceptance/submission should be included.

3. The individual articles may have been written together with the main supervisor, another supervisor or other persons. In order to show that the candidate has attained the intended proficiency, the majority of the articles must have been written by the candidate personally (which should normally result in first authorship) and comprise a substantial part of the thesis as a whole. The thesis should be accompanied by a detailed description of all the authors' contributions to each of the articles.

4. In terms of its scholarly contribution, a cumulative thesis in its entirety shall satisfy the same academic requirements as a thesis in the form of a monograph. Any submission of constituent parts that had been published/submitted/written before the student started his/her doctoral/masters studies or is primarily based on prior work may be included to contextualize the thesis if properly declared, but typically does not count towards the scholarly contribution of the thesis.

5. Before your article is published by a publisher, you can attempt to retain specific rights to your work through a publication agreement addendum, such as the SPARC Canadian Author Addendum. If you did not retain rights to re-use your article, you can request permission to include your article in your thesis by emailing or writing to the copyright holder, explaining how and why you want to use the work and requesting permission. If granted permission, you should keep a record of who gave the permission, what was permitted, the date, and how to contact the person who gave the permission. A copy of each relevant copyright release must be included in your Thesis Package. If the publisher will not grant you permission, it may still be possible to use the content of the pre-print or post-print of your article, depending on the publisher's copyright policies outlined in the publisher's copyright transfer / author publication agreement, many of which can be found on the SHERPA/RoMEO website. You can find the latest information about copyright on the Copyright at SFU website. If you have questions about retaining or obtaining copyright permission, you can always contact your liaison librarian.

Appendix B

The Significance of the Low Complexity Dimension in Music Similarity Judgements

As published in Ens, J. & Pasquier, P. (2017). *The Significance of the Low Complexity Dimension in Music Similarity Judgements.* ISMIR.

Abstract

Previous research has demonstrated that similarity judgements are context specific, as they are shaped by cultural exposure, familiarity, and the musical aesthetic of the content being compared. Although such research suggests that the criterion for similarity judgement varies with respect to the musical style of the content being compared, the specific musical factors which shape this criterion are unknown. Since dimensional complexity differentiates musical genres, and has been shown to affect similarity judgements following lifelong exposure, this experiment investigates the short-term influence of dimensional complexity on similarity judgements. Rhythmic and pitch sequences with two levels of complexity were factorially combined to create four distinct types of prototype melodies. 51 participants rated the similarity of each type of prototype melody (M) to two variations, one in which the pitch content was modified $(\bar{M_p})$, and another in which the rhythmic content was modified $(\bar{M_r})$. The results indicate that rhythm and pitch complexity both play a significant role, influencing the perceived similarity of $\bar{M_p}$, and $\bar{M_r}$. The dimension bearing low complexity information was found to be the predominant factor in similarity judgements, as participants found modifications to this dimension to significantly decrease perceived similarity.

Keywords: Generative Music; Evaluation; Machine Learning; Big Data

B.1 Introduction

Similarity directly informs our experience of music, enabling the perception of cohesion within a musical work, and the categorization of musical works. Consequently, developing models that encapsulate the manner in which similarity is perceived, is of critical importance within the areas of Musicology, Music Cognition and Music Theory [30]. In particular, the search for robust and flexible similarity measures has dominated research in the Music Information Retrieval (MIR) domain, as large digital databases of music information necessitate content-based querying and retrieval, and classification. Although there is a large body of research that explores similarity perception within music, many aspects of similarity perception are not yet fully understood. The current study corroborates previous evidence that similarity criterion vary with respect to the musical content being compared [9], demonstrating that the complexity of pitch and rhythmic content influence similarity perception.

Since pitch and rhythm are the two most prominent musical dimensions in the context of symbolic notation, the current study will manipulate complexity along these dimensions and observe the effects on similarity perception. Although no musical dimensions are completely orthogonal, as a modification in a particular dimension may affect the perception of other dimensions, the complexity of pitch and rhythmic content can be measured independently, and there is evidence that these dimensions are processed separately in cognition [13, 27]. Therefore, pitch and rhythm complexity were considered to be independent for the purposes of this study. *Pitch content* refers to the sequence of pitches encapsulated in a particular melody, and *rhythm content* refers to the sequence of durations. *Dimensional complexity* refers to the absolute level of complexity along a particular musical dimension. In this study we measure the dimensional complexity of pitch and rhythm content.

B.2 Related Work

Previous work examining the perception of musical similarity, has focused on establishing a hierarchy of musical dimensions, ranking their observed contributions to similarity perception. On a whole, most research claims that rhythmic information is the most important. Halpern [7] constructed 16 melodies — a factorial combination of two pitch sequences, two rhythmic sequences, two tonal structures and forward and reversed versions — and found that rhythm was the most important distinguishing factor, followed by pitch, direction and tonal structure. Similarly, Rosner and Meyer [19] found rhythm to be the strongest determinant of melodic similarity. Despite the general consensus that rhythm plays a dominant role in similarity judgements, pitch still plays a considerable role. Dowling [2] demonstrated that a modified imitation of a prototype melody is often misidentified as the prototype when it has a similar pitch contour.

Given the multidimensional nature of music, many researchers have found it useful to make the distinction between surface-level and structural features. In general, surface-level attributes include contour, loudness and tempo while structural attributes denote aspects of form, thematic development and patterns. In short term contexts, where participants are unfamiliar with the musical material being compared, surface-level features are a strong predictor of both melodic [19, 15, 22] and polyphonic [9] similarity. Prince [15] found that rhythm was the dominant aspect informing perceived melodic similarity, followed by contour, meter, and tonal structure.

However, there is increasing evidence which questions the generality of these results, as contextual factors including familiarity, cultural exposure, and the aesthetic of the musical content being compared, have been shown to have a considerable effect on similarity perception. Pollard-Gott found that with repeated listening, surface level features became less influential and thematic material became more important [14]. Similarly, the long term analysis of a collection of folk melodies by a panel of experts, placed emphasis on thematic and motivic similarity above all other factors [31]. Schubert and Stevens [22] found that contour is more important than harmonic structure for making similarity comparisons, but with musical expertise, harmonic structure also has an effect.

Other research has shown that cultural exposure affects similarity perception. Hannon and Trehub [8] found the metrical bias of North American adults to be the result of an enculturation processes, with no evidence of a natural predisposition for the simple meters which characterize much of western music. Goldstone [6] suggests that humans learn by focusing on perceptual features that are more informative, at the cost of decreased attention towards other dimensions. This phenomenon has been observed in a musical context, where the voice that consists of immediate and exact repetitions of a short musical fragment tends to perceptually decrease in salience for the listener over time [24]. Instead, the listener is naturally drawn to focus on the high complexity voice. Since distinct rhythmic durations occur at a relatively higher frequency than distinct pitches in western music, they demand less attention than pitch content. After years of exposure, this likely results in an increased sensitivity to the pitch content in a melody [17]. Notably, Eerola et al. [3] demonstrated that musical complexity perceptions are shaped by exposure to different musical culture, which likely results from the mechanisms described above.

In addition to the factors mentioned above, music aesthetic has been shown to influence how similarity is perceived. Lamont and Dibben [9] examined similarity relationships in two contrasting musical styles, requiring participants to rate the similarity of extracts from a Beethoven sonata (op. 10, no. 1, first movement) and a dodecaphonic work composed by Schoenberg (Klavierstück op. 33a). Nine polyphonic excerpts were selected from each piece, each approximately eight measures long, and the similarity of each possible combination was rated by participants, resulting in 36 similarity ratings for each piece. Notably, both pieces are composed for solo piano, and have more than one theme which is developed throughout the duration of each work. They found that similarity judgements were primarily based on surface level features, however, the similarity judgements for each piece were predominantly influenced by different surface features. These results suggested that each piece establishes a different similarity criterion within which listeners make appropriate similarity judgements. Although Lamont and Dibben demonstrated that the criterion for similarity judgements varies with respect to the musical aesthetic of the stimuli being compared, the specific musical factors which caused this phenomenon are still unknown, directly motivating our experiment.

B.3 Motivation

As evidenced by the brief overview in section B.2, numerous studies have demonstrated the prevalent influence of contextual factors on musical similarity judgements [14, 31, 8, 17, 9], directly motivating further study in this area. Since contextual factors like cultural exposure and familiarity are difficult to integrate into a similarity measure, this study examines the third contextual factor, the role of the musical content itself in shaping a criterion for similarity judgements. The phenomenon that Lamont and Dibben [9] observed, provides evidence that musical content influences the manner in which music is compared, as participants used different musical dimensions to make comparisons depending on the nature of the musical content. In light of this evidence, it is worthwhile to examine how specific musical characteristics of the content being compared shape similarity judgements, which does not appear to have been examined previously. Due to the fact that dimensional complexity differentiates musical genres [3], and affects similarity judgements following lifelong exposure [8], this experiment investigates the short-term influence of dimensional complexity on melodic similarity judgements. More specifically, this study investigates the role of dimensional complexity in shaping awareness to modifications in that particular dimension, effectively establishing a criterion for melodic similarity judgements.

Previous research has shown that limitations on the human capacity for musical memory, have an effect on musical perception. Participants found it more difficult to retain melodies with complex contours, which were devoid of any repetition, and were often unable to distinguish them from another complex contour [18]. Moreover, complexity was one of four variables which collectively predicted the recognizability of melodies when presented a second time [20]. In these cases, it seems likely that working memory limitations make it difficult to encapsulate all aspects of a complex melody on first exposure. In summarizing recent research on working memory limitations, Cowan [1] proposes that there is a capacity of three to five chunks in working memory for young adults. According to these findings, modifications to the musical dimension bearing the least complex musical material should be the easiest to detect, which suggests that this musical dimension would have a predominant influence on similarity judgements. Collectively, this research supports the following hypothesis: modifications to the musical dimension bearing low complexity information will result in a significant decrease in similarity, in comparison to similar modifications to the musical dimension bearing high complexity information.

B.4 Methodology

B.4.1 Participants

The participants were recruited online using the Crowdflower [1] crowdsourcing platform, and required to pass a test before participating in the experiment. Participants were paid \$0.02 USD for each question they answered, in accordance with the typical compensation offered to Crowdflower users. Of the 96 participants who took the test, 76 passed (79.2%) and 63 completed the experiment. 12 participants responses were deemed ineligible based on the inconsistent responses to an identical question. In total, 51 participants came from 25 different countries.

B.4.2 Stimuli

Measuring Complexity

Given the multifaceted nature of complexity, it is necessary to make the distinction between the entropy based complexity measures proposed by Eerola et al. [3], and the notion of complexity

[1] https://www.crowdflower.com/

which grounds the current study. Shannon Entropy quantifies the disorder or uncertainty inherent in an information source based on a representative probability distribution [23]. Eerola et al. calculate entropy using the marginal probability of each symbol in a sequence. This type of complexity will be referred to as *entropy*$_m$. Although entropy$_m$ has been shown to correlate with the percieved complexity of musical sequences [16], this measurement of complexity does not provide the necessary resolution to make comparisons between many musical sequences. For an explicit example, consider the following pitch sequences, $s_1 = \{$c, d, e, f, c, d, e, f$\}$, and $s_2 = \{$c, f, e, d, e, c, d, f$\}$. Even though s_1 exhibits less complexity than s_2, both s_1 and s_2 have the same entropy$_m$, as this measurement does not take the repetition of longer phrases into consideration. Clearly, it is necessary to take the repetition of phrases into consideration when measuring complexity.

Admittedly, this can be accomplished by calculating the entropy rate of an n-th order markov chain derived from the musical sequence being measured, however there are still issues with this approach. In contrast to the manner in which humans perceive musical content, and by extension musical complexity, the entropy rate is not designed to distinguish between repetition which occurs within the prevailing metric structure, and repetition which spans metrical boundaries. Research suggests that humans perceive music by breaking it into a series of chunks [5], and have a natural tendency to project metre onto sequences of sound, despite the absence of acoustic cues for metric organization [4]. In addition, when listening to music, humans naturally extract motivic patterns [32], and larger formal structures [12]. Since humans segment music in accordance with metrical boundaries, it is likely that humans are less sensitive to repetition which is obscured by these boundaries. Consequently, a true measure of musical complexity must take this distinction into account.

Furthermore, an entropy based model of complexity is not capable of taking similarity into consideration, as entropy is based on the lossless encoding of an information source [23]. This becomes more of an issue when entropy is being measured with respect to larger subsequences, as is the case when measuring the n-th order entropy rate. This formulation of complexity cannot make the distinction between a collection of subsequences which share the same contour, and a collection that does not. As a result, it seems most reasonable to take the collective dissimilarity of subsequences segmented with respect to the prevailing metric structure, as a measure of complexity. Consequently, a homogeneous collection of segments would be perceived as having a low complexity, while a diverse collection of segments would be perceived as having a high complexity. We use the term *redundancy* to refer to this type of complexity throughout the paper.

In order to quantify redundancy, two different measures were used. Thul's [28] adaptation of Tanguiane's [26, 25] algorithm, measures redundancy by counting the number of *root patterns*, at several hierarchical levels. This will be referred to as *Tanguiane's Rhythmic Complexity* (*TRC*). The other measure of redundancy is calculated using Eqn (B.1), where (S) is a set of subsequences, derived by segmenting a sequence of symbols into measures. Notably, Eqn (B.1) also requires a distance metric (D). Chronotonic distance [29] is used to measure *Rhythmic Sequence Complexity* (*RSC*), and a similarity measure proposed by Maidín [10] is used to measure *Pitch Sequence Complexity* (*PSC*). Admittedly, segmenting a pitch sequence according to metre means that *PSC* is dependant on the rhythmic content, however, within-measure rhythmic patterns have no bearing on *PSC* in this paradigm, and the metric structure is not being manipulated in this study. Although *PSC* does not account for the complexity of individual segments, section B.4.2 describes how complexity is restricted in this experiment, effectively mitigating the variance of segment complexity in the current study.

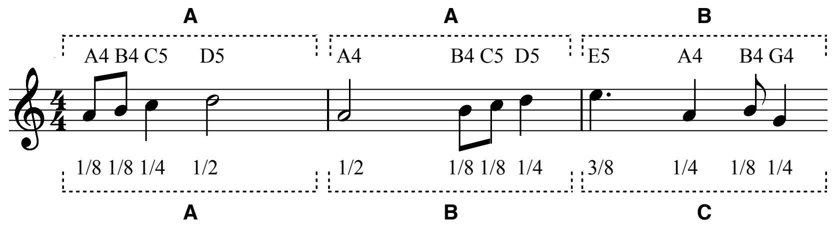


Figure B.1: A melody with complex rhythm and simple pitch, using letters to show the form of each dimension.

$$f(S) = \frac{1}{|S|} \sum_{i=1}^{|S|} \min\{D(S_i, S_j) : j \neq i; 1 \leq j \leq |S|\} \tag{B.1}$$

Prototype Melodies

In this experiment, there were four types of melodies; *rhythm*$_s$-*pitch*$_s$, *rhythm*$_s$-*pitch*$_c$, *rhythm*$_c$-*pitch*$_s$, and *rhythm*$_c$-*pitch*$_c$, where s denotes a simple or low complexity sequence, and c denotes a complex sequence [2]. In addition, eight versions of each melody type were constructed, resulting in 32 (4×8) prototype melodies of equal length (three measures). As mentioned in section B.4.2, redundancy quantifies the degree to which an information source is self similar and contains periodic repetition in conjunction with the prevailing metrical structure. In light of this aim, melodies were comprised of three measure-length phrases, with phrase repetition varied to create two distinct levels of complexity. Low complexity sequences had a formal pattern AAB, where a pattern is repeated in the first two measures, and a new pattern is introduced in the last measure. High complexity sequences had a formal pattern ABC, where each measure is dissimilar. This construction process is demonstrated in Figure B.1, which shows a high complexity rhythm sequence and a low complexity pitch sequence.

Care was taken to restrict the variability of entropy$_m$ based complexity, using measures proposed by Eerola et al. [3]. Since the pitch sequences were constructed from scales consisting of five distinct pitch classes, *Entropy of pitch class distribution* and *Entropy of interval distribution* did not vary significantly. Similarly, rhythm sequences were constructed from four distinct durations, limiting the variance of *Entropy of note duration distribution* and *Rhythmic variability*. Notably, it seemed reasonable to have fewer distinct durations than pitch classes, as research has demonstrated that most listeners are able to perceive pitch diversity more readily [17]. A One-Way Analysis of Variance (ANOVA) across all four prototype melody types demonstrated that none of these entropy$_m$ based complexity measures were a significant source of variance, while *PSC*, *RSC* and *TRC* varied significantly. Furthermore, the entropy rate – calculated using a first order markov chain – did not vary significantly across melody type. This verified that our experiment measured the effect of variations in redundancy in relative isolation.

In order to restrict the variance of segment complexity, *Mean interval size* and *Note density* were restricted, which Eerola et al. [3] found to be a significant source of complexity. Each melody was constrained to an octave range, restricting the *Mean interval size*. The *Note density*, was invariant

[2]The melodies used in this experiment can be found at https://mlab-experiments.iat.sfu.ca/ismir2017/audio

for each constructed melody, as each melody had four notes per measure, and was three measures long.

Modified Melodies

For each prototype melody (M), two modified versions were constructed for the main experiment: a version in which the pitch is modified $(\bar{M}_p)$, and a version in which the rhythm is modified $(\bar{M}_r)$. This process involved reversing the order of the measures in the dimension which is to be modified. As a result, regardless of the nature of the prototype melody, the first and last measures of the modified melody were different. Since test questions required a ground truth answer, three additional types of modified melodies were constructed: a melody in which the pattern form of M was transformed from AAB to ABA in the pitch dimension $(\bar{M}_{r\bar{p}})$, a melody in which the pattern form of M was transformed from AAB to ABA in the rhythm dimension $(\bar{M}_{p\bar{r}})$, and a melody in which both dimensions were modified $(\bar{M}_b)$.

B.4.3 Experimental Design

The experiment consisted of two independent variables, rhythm and pitch content complexity. Both rhythm and pitch complexity had two levels, low and high. This resulted in a 2×2 repeated measures experimental design, with four distinct types of prototype melodies. Participants were presented with a series of questions, consisting of a prototype melody (M) and two modified melodies (*melody A*, *melody B*). There were two types of test questions, which were developed using the modified melodies described above. The first type of question, compared either $\bar{M}_{r\bar{p}}$ and M against the prototype M, or $\bar{M}_{p\bar{r}}$ and M against M. This had an indisputable answer, as one of the modified melodies was in fact an exact replica of the prototype. The second type of question, compared $\bar{M}_p$ and $\bar{M}_b$ to the prototype, or compared $\bar{M}_r$ and $\bar{M}_b$ to the prototype. Given the manner in which these melodies were constructed, $\bar{M}_p$ and $\bar{M}_r$ are more similar to the prototype, as they are identical to the prototype along a single dimension, while $\bar{M}_b$ is dissimilar in both dimensions.

For the actual experiment itself, there was a single type of question, in which $\bar{M}_r$ and $\bar{M}_p$ were compared against the prototype. Irregardless of the type of question, the two modified melodies were randomly assigned to be *melody A* or *melody B*. For each question, participants rated the similarity of *melody A* to M, and *melody B* to M, on a Likert scale from 1 to 20, where 20 indicates maximal similarity. In the analysis below, the difference $(D = S(M, \bar{M}_r) - S(M, \bar{M}_p))$ between the perceived similarity of $\bar{M}_r$ to M $(S(M, \bar{M}_r))$, and the perceived similarity of $\bar{M}_p$ to M $(S(M, \bar{M}_p))$, is taken as the dependent variable. As a result, a positive value of D indicates that modifications to the rhythm dimension have less of an effect on similarity than modifications to the pitch dimension, while a negative value of D indicates the opposite.

B.4.4 Procedure

Before participating in the experiment, participants were required to complete 10 test questions with a minimum accuracy of 80%. The test questions served two purposes, eliminating those who were not taking the task seriously, and familiarizing participants with the similarity domain within which

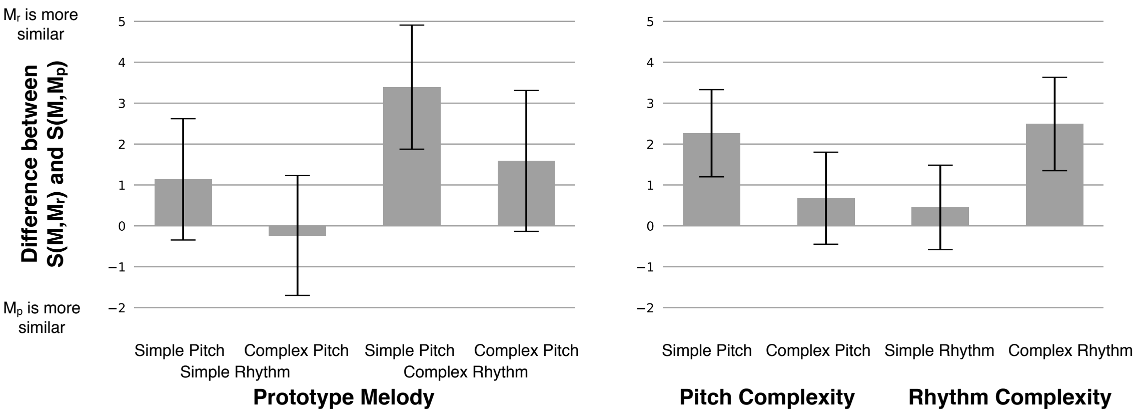


Figure B.2: (a) The difference between the perceived similarity of the modified rhythm melody and the perceived similarity of the modified pitch version for each prototype melody complexity category, with 95% confidence intervals. (b) The main effects of pitch and rhythm complexity with 95% confidence intervals

they were being asked to make comparisons. Once the test was successfully completed, participants were presented with 10 randomly ordered questions, consisting of eight different experiment questions (representing each of the eight different types of prototype melodies), a test question, and a repeated experiment question. The repeated experiment question was used to determine if participants were answering the questions consistently. For each question, the prototype melody was selected randomly from a collection of eight versions, and the key was randomly transposed so that the content varied from question to question. After listening to all three melodies, participants were asked to indicate which of the two modified versions was more similar to the prototype, and rate the similarity of *melody A* and *melody B* on a Likert scale from 1 to 20.

B.5 Results

Since the ANOVA is relatively robust to violations of normality [21], the 2-Way ANOVA was conducted without transforming the data, despite the violation of the assumption of normality. A 2-Way ANOVA revealed the main effect of rhythm complexity $(F(50) = 9.17, p = .004, \eta_p^2 = .155)$ and pitch complexity $(F(50) = 5.31, p = .025, \eta_p^2 = .096)$, while the interaction between rhythm complexity and pitch complexity was insignificant $(p = .657)$. To be thorough, an Aligned Rank Transform was performed on the data, correcting for the effects of the non-normal distributions of the data [33]. Using the transformed data, a 2-Way ANOVA revealed main effect of rhythm complexity $(F(50) = 9.82, p = .003, \eta_p^2 = .164)$ and pitch complexity $(F(50) = 6.26, p = .016, \eta_p^2 = .111)$, while the interaction between rhythm complexity and pitch complexity was insignificant $(p = .601)$. These results corroborate the analysis of the untransformed data, indicating that 16.4% of the variability in similarity ratings were explained by changes in rhythm complexity, and 11.1% of the variability was explained by changes in pitch complexity.

As predicted, there was a main effect of rhythm complexity and pitch complexity, both shown in Figure B.2. Melodies containing low complexity rhythmic content $(M = 0.451, SD = 5.26)$ were significantly lower than those containing high complexity rhythmic content $(M = 2.49, SD = 5.81)$, which indicates that participants were more sensitive to pitch modifications when pitch sequences were less complex. This effect was pronounced in cases where the rhythmic sequence was more complex, as participants found pitch modified melodies $(\bar{M_p})$ to be significantly less similar to *rhythm*$_c$-*pitch*$_s$ prototype melodies than rhythm modified melodies $(\bar{M_r})$. Conversely, melodies containing low complexity pitch content $(M = 2.26,\ SD = 5.43)$ were significantly higher than those containing high complexity pitch content $(M = 0.676,\ SD = 5.73)$, which indicates that participants were more sensitive to rhythmic modifications when rhythmic sequences were less complex. Similarly, this effect was pronounced in cases where the pitch sequence was more complex, as participants found rhythm modified melodies $(\bar{M_r})$ to be significantly less similar to *rhythm*$_s$-*pitch*$_c$ prototype melodies than pitch modified melodies $(\bar{M_p})$. Therefore, the dimension bearing low complexity musical content was found to play a significant role in similarity judgements, as modifications to that dimension significantly decreased perceived similarity.

An analysis of the individual prototype melody conditions revealed that the *rhythm*$_s$-*pitch*$_c$ condition $(M = -0.235,\ SD = 5.21)$ was significantly less than the *rhythm*$_c$-*pitch*$_s$ condition $(M = 3.39,\ SD = 5.39)$, as pitch modified melodies were the most similar to *rhythm*$_s$-*pitch*$_c$ prototypes, and rhythm modified melodies were the most similar to *rhythm*$_c$-*pitch*$_s$ prototypes. The *rhythm*$_s$-*pitch*$_s$ condition $(M = 1.14,\ SD = 5.27)$ and the *rhythm*$_c$-*pitch*$_c$ condition $(M = 1.59,\ SD = 6.12)$ were roughly equivalent, and participants did not find a particular type of modified melody to be more similar, relative to the two other conditions. Collectively, these results indicate that melodies which are modified in the dimension bearing low complexity information are perceived as significantly less similar than melodies which are modified in the dimension bearing high complexity information.

B.6 Discussion

As evidenced by the results presented above, modifications to the dimension bearing low complexity information result in a significant decrease in perceived similarity, demonstrating that the dimension bearing low complexity information plays a more significant role in melodic similarity judgments. On a whole, the values for all four conditions were positively skewed (Figure 2a), indicating that modifications to the pitch content of a melody had a greater influence on perceived similarity. Since there is no benchmark with which to compare rhythmic sequence complexity and pitch sequence complexity, it was not possible to equate the complexity across dimensions. Consequently, some skew in either direction was expected. The positive skew may indicate that the rhythmic content of the melodies in this experiment was on average more complex, and participants had difficulty noticing modifications in the rhythm dimension. Alternatively, due to the enculturation process that Hannon and Trehub [8] observed, participants may have paid more attention to the pitch content, resulting in the slight positive skew. When these factors are considered, it is arguably most meaningful to interpret the conditions in relation to each other, as some skew in either direction was inevitable. Viewed from this perspective, the hypothesis is directly corroborated, as the *rhythm*$_s$-*pitch*$_c$ condition is the lowest, the *rhythm*$_c$-*pitch*$_s$ is the highest, and the *rhythm*$_s$-*pitch*$_s$ and *rhythm*$_c$-*pitch*$_c$ conditions are in the middle.

Further analysis reveals that previous experiments are likely a special case of the generalized theory proposed in this paper. Monahan et al. [11] and Halpern [7] both make the claim that rhythm contributes more significantly to similarity perception, however, the rhythmic component of their stimuli is predominantly low complexity, and the pitch component of their stimuli is relatively higher on average. Notably, this was measured using *PSC*, *RSC*, and *TRC*. Although Halpern and Monahan et al. attribute their results to an inherent bias towards rhythm, the results of this experiment suggest that the relative complexity of the rhythm and pitch content provides a more robust explanation.

Admittedly, there are several limitations to the generalization of the results of this study. First and foremost, the observed relationship between dimensional complexity and similarity judgements may manifest itself quite differently when working with longer melodies, or polyphonic music. Secondly, due to the fact that musical complexity is multifaceted and far from understood, determining the relatively low complexity dimension may be quite difficult in some contexts. Despite the aforementioned limitations, the limited variance of Eerola et al.'s entropy based complexity measures provides substantial support for the generalization of these findings, as most western music makes use of the same limited collection of distinct note durations and pitch classes [16]. As a result, although this form of entropy based complexity is the source of some variability within the musical cannon, redundancy arguably accounts for more of this variation. Consequently, the results of this study are not restricted to a particular genre, and are relevant across musical genres.

B.7 Conclusion

Similarity is shaped by several factors, including familiarity, and cultural conditioning. This study asserts the significance of another factor – the nature of the musical content which is being compared – by examining the effects of dimensional complexity on similarity judgements. The general notion that characteristics of the musical content being compared have some bearing on the criterion used to make similarity judgements, is not new, and has been observed in past experiments [9]. However, the manner in which musical content establishes a criterion for similarity judgements has not been explored previously. The results of this study provide evidence that pitch and rhythmic complexity are factors which shape the criterion used in similarity judgements, as the dimension bearing relatively low complexity information has a greater influence on similarity perception. Furthermore, the results of this experiment are corroborated by previous experiments [7, 11], offering a general explanation for these previous findings.

Developing robust and flexible similarity measures continues to be a dominant area of research in the MIR domain, as large digital databases of music information necessitate accurate methods for comparison and categorization. As a result, adapting existing similarity measures to take dimensional complexity into account, is a possible application of the findings of this study. Future research is also necessary to investigate the role of complexity along other dimensions, including dynamics, articulation and timbre. Furthermore, the manner in which complexity is perceived along a single dimension is in need of continued exploration, as several issues with pre–existing methods for measuring complexity have been discussed in section B.4.2. Clearly, musical similarity is a complex phenomenon which is deserving of continued exploration, as the results of this experiment have explicitly demonstrated that similarity judgements are dependant on another contextual factor, the complexity of pitch and rhythm content in the musical material being compared.

Bibliography

[1] Nelson Cowan. “The Magical Mystery Four: How Is Working Memory Capacity Limited, and Why?” In: *Current Directions in Psychological Science* 19.1 (2010), pp. 51–57.

[2] Walter J. Dowling. “Scale and contour: Two components of a theory of memory for melodies.” In: *Psychological Review* 85.4 (1978), pp. 341–354.

[3] Tuomas Eerola, Tommi Himberg, Petri Toiviainen, and Jukka Louhivuori. “Perceived complexity of western and African folk melodies by western and African listeners”. In: *Psychology of Music* 34.3 (2006), pp. 337–371.

[4] Paul Fraisse. “Rhythm and Tempo”. In: *The Psychology of Music*. Ed. by Diana Deutsch. New York: Academic Press, 1982, pp. 149–181.

[5] Rolf Inge Godøy, Alexander Refsum Jensenius, and Kristian Nymoen. “Chunking in music by coarticulation”. In: *Acta Acustica united with Acustica* 96.4 (2010), pp. 690–700.

[6] Robert L. Goldstone. “Learning to perceive while perceiving to learn”. In: *Perceptual Organization in Vision: Behavioural and Neural Perspectives*. Ed. by R Kimchi, M Behrmann, and C Olson. New Jersey: Lawrence Erlbaum Associates, 2003, pp. 233–278.

[7] Andrea R. Halpern. “Perception of structure in novel music”. In: *Memory & cognition* 12.2 (1984), pp. 163–170.

[8] Erin E Hannon and Sandra E Trehub. “Metrical categories in infancy and adulthood”. In: *Psychological Science* 16.1 (2005), pp. 48–55.

[9] Alexandra Lamont and Nicola Dibben. “Motivic Structure and the Perception of Similarity”. In: *Music Perception: An Interdisciplinary Journal* 18.3 (2001), pp. 245–274.

[10] Donncha O. Maidín. “A geometrical algorithm for melodic difference”. In: *Computing in musicology: a directory of research* 11 (1998), pp. 65–72.

[11] Caroline B Monahan and Edward C Carterette. “Pitch and Duration as Determinants of Musical Space”. In: *Music Perception: An Interdisciplinary Journal* 3.1 (1985), pp. 1–32.

[12] Christiane Neuhaus, Thomas R Knösche, and Angela D Friederici. “Similarity and repetition: An ERP study on musical form perception”. In: *Annals of the New York Academy of Sciences* 1169 (2009), pp. 485–489.

[13] Isabelle Peretz and Max Coltheart. “Modularity of music processing.” In: *Nature neuroscience* 6.7 (2003), pp. 688–691.

[14] Lucy Pollard-Gott. “Emergence of Thematic Concepts in Repeated Listening to Music”. In: *Cognitive psychology* 15.1 (1983), pp. 66–94.

[15] Jon B. Prince. “Contributions of pitch contour, tonality, rhythm, and meter to melodic similarity.” In: *Journal of experimental psychology. Human perception and performance* 40.6 (2014), pp. 2319–37.

[16] Jon B. Prince and Peter Q. Pfordresher. “The role of pitch and temporal diversity in the perception and production of musical sequences”. In: *Acta Psychologica* 141.2 (2012), pp. 184–198.

[17] Jon B. Prince, Mark A. Schmuckler, and William F. Thompson. “The effect of task and pitch structure on pitch-time interactions in music”. In: *Memory & cognition* 37.3 (2009), pp. 368–381.

[18] Thomas W. Reiner. “Pitch-distance and contour complexity in the recognition of short melodies”. In: *Journal of Scientific Psychology* September (2011), pp. 27–36.

[19] Burton S. Rosner and Leonard B. Meyer. “The perceptual roles of melodic process, contour, and form”. In: *Music Perception* 4.1 (1986), pp. 1–39.

[20] P. A. Russell. “Memory for music: A study of musical and listener factors”. In: *British Journal of Psychology* 78.3 (1987), pp. 335–347.

[21] Emanuel Schmider, Matthias Ziegler, Erik Danay, Luzi Beyer, and Markus Bühner. “Is It Really Robust?: Reinvestigating the robustness of ANOVA against violations of the normal distribution assumption”. In: *Methodology* 6.4 (2010), pp. 147–151.

[22] Emery Schubert and Catherine Stevens. “The effect of implied harmony, contour and musical expertise on judgments of similarity of familiar melodies”. In: *Journal of New Music Research* 35.2 (2006), pp. 161–174.

[23] Claude Elwood Shannon. “A Mathematical Theory of Communication”. In: *The Bell System Technical Journal* 27.July (1948), pp. 379–423.

[24] Cecilia Taher, René Rusch, and Stephen McAdams. “Effects of Repetition on Attention in Two-Part Counterpoint”. In: *Music Perception* 33.3 (2016), pp. 306–318.

[25] Andranick S. Tanguiane. “A Principle of Correlativity of Perception and Its Application to Music Recognition”. In: *Music Perception: An Interdisciplinary Journal* 11.4 (1994), pp. 465–502.

[26] Andranick S. Tanguiane. *Artificial Perception and Music Recognition*. Berlin: Springer-Verlag, 1993, pp. 1–210.

[27] William Forde Thompson, Michael D Hall, and Jeff Pressing. “Illusory conjunctions of pitch and duration in unfamiliar tone sequences”. In: *J Exp Psychol Hum Percept Perform* 27.1 (2001), pp. 128–140.

[28] Eric Thul. “Measuring the Complexity of Musical Rhythm”. PhD thesis. McGill University, 2008.

[29] Godfried T. Toussaint. “A comparison of rhythmic similarity measures”. In: *Proc. International Conference on Music Information Retrieval (ISMIR 2004)* (2004), pp. 242–245.

[30] Anja Volk, Elaine Chew, Elizabeth Hellmuth Margulis, and Christina Anagnostopoulou. “Music Similarity: Concepts, Cognition and Computation”. In: *Journal of New Music Research* 45.3 (2016), pp. 207–209.

[31] Anja Volk and Peter Kranenburg. “Melodic similarity among folk songs: An annotation study on similarity-based categorization in music”. In: *Musicae Scientiae* 16.3 (2012), pp. 1–23.

[32] Robert L. Welker. “Abstraction of Themes from Melodic Variations”. In: *Journal of Experimental Psychology. Human Perception and Performance* 8.3 (1982), pp. 435–447.

[33] Jacob O Wobbrock, Leah Findlater, Darren Gergle, and James J Higgins. “The aligned rank transform for nonparametric factorial analyses using only ANOVA procedures”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2011), pp. 143–146.

Appendix C

Discriminating Symbolic Continuations with GenDetect

As published in Ens, J. & Pasquier, P. (2019). *Discriminating Symbolic Continuations with GenDetect.* MIREX 2019 Results.

Abstract

We describe GenDetect, an algorithm that was submitted to the 2019 MIREX Patterns for Prediction task. GenDetect is used to discriminate between two possible continuations of a prime, distinguishing a genuine continuation from a generated one. Each musical excerpt is represented by a collection of categorical distributions and a Gradient Boosting Classifier is trained to predict the genuine continuation using this representation. Two versions of the algorithm were submitted, one for polyphonic music and another for monophonic music.

Keywords: Generative Music; Evaluation; Machine Learning; Big Data

C.1 Introduction

The 2019 MIREX Patterns for Prediction task consists of two sub-tasks. GenDetect is designed for the second sub-task, which involves discriminating between two continuations $(\mathbb{M}_a, \mathbb{M}_b)$ of a prime $\mathbb{M}_{\text{prime}}$. In contrast to *BachProp* [1], which ranks $\mathbb{M}_a$ and $\mathbb{M}_b)$ according to their probability under a recurrent neural network that is trained to model musical material, we train a Gradient Boosting Classifier [2] to predict the genuine continuation given a feature-based representation of $\mathbb{M}_a, \mathbb{M}_b$ and $\mathbb{M}_{\text{prime}}$. We build on the data representation used by *StyleRank*, representing each musical excerpt (e.g. $\mathbb{M}_a$) with a collection of categorical distributions [3].

C.2 Methodology

In what follows we adopt the following notation. Given a set x, $||x||$ denotes the number of elements in the set x, and x_i denotes the i^{th} element in x (1-indexed). $\max(x)$ and $\min(x)$ denote the maximum and minimum elements in x respectively. $\ll$ indicates a left bitwise shift. $\mathbf{I}(\cdot)$ is a function that returns 1 if the predicate $\cdot$ is true and 0 otherwise. Let $\oplus$ denote the concatenation operation.

C.2.1 Data Representation

Consider a musical excerpt $\mathbb{M} = [m_1, ..., m_n]$, consisting of n notes (m_i) ordered lexicographically, sorting first by onset and then by pitch height. Let $\mathbb{P} = [\texttt{pitch}(m_i) : 1 \leq i \leq n]$ [1], $\mathbb{O} = [\texttt{qnt}(\texttt{ons}(m_i)) : 1 \leq i \leq n]$, and $\mathbb{D} = [\texttt{qnt}(\texttt{dur}(m_i)) : 1 \leq i \leq n]$, where `pitch`, `ons` and `dur` are functions returning the pitch, onset and duration of a note respectively. `qnt` refers to Eq. (C.1), which accepts time-based values (e.g. onset and duration) and returns an integer rounded to the nearest r^{th} subdivision of a beat.

$$\texttt{qnt}(x) = \lceil xr - 0.5 \rceil \tag{C.1}$$

The following procedure is applied to segment $\mathbb{M}$ into chords, where $\texttt{off}(m_i) = \texttt{qnt}(\texttt{ons}(m_i) + \texttt{dur}(m_i))$. First we construct two sets, one containing all unique note onsets $\mathbb{B}_{\text{onset}} = \{\texttt{ons}(m_i) : m_i \in \mathbb{M}\}$ and another containing all unique note offsets $\mathbb{B}_{\text{offset}} = \{\texttt{off}(m_i) : m_i \in \mathbb{M}\}$. Then we construct the ordered set $\mathbb{B} = \mathbb{B}_{\text{onset}} \cup \mathbb{B}_{\text{offset}}$, where the elements are arranged in ascending order. The i^{th} chord is the set of notes that completely overlap the interval $[\mathbb{B}_i, \mathbb{B}_{i+1}]$, and can be calculated using Eq. (C.2). As a result, there are $||\mathbb{B}|| - 1$ chords in $\mathbb{M}$, and rests are equivalent to chords containing no notes $(\mathbb{C}_i = \emptyset)$. In what follows, let $\mathbb{C}_i^j$ denote the j^{th} note in the i^{th} chord, and $\psi(\mathbb{C}_i) = \{\texttt{pitch}(\mathbb{C}_i^j) : \mathbb{C}_i^j \in \mathbb{C}_i\}$. In addition , we sort the notes in each chord in ascending order according to pitch height.

$$\mathbb{C}_i = \{n : (n \in \mathbb{M}) \wedge (\texttt{ons}(n) \leq \mathbb{B}_i) \wedge (\texttt{off}(n) \geq \mathbb{B}_{i+1})\} \tag{C.2}$$

[1]Note that we adapt the set-builder notation to construct a list (e.g., $[i/2 : 0 \leq i < 4] = [0, 0, 1, 1]$), which unlike a set, may contain duplicate values, and has a specific order.

	Feature Name	Function	Domain
Mono.	Chord Size $\star$	$\sum_{i=1}^{\lVert\mathbb{B}\rVert-1} \mathbf{I}_k(\lVert\mathbb{C}_i\rVert)(\mathbb{B}_{i+1} - \mathbb{B}_i)$	$[0, 2)$
	Melodic n-gram PCD	$\sum_{i=1}^{\lVert\mathbb{P}\rVert-w+1} \mathbf{I}_k(\texttt{PCD}(\{\mathbb{P}_j \mod 12 : i \leq j < i + w\}))$	$[0, 352)$
Both	Note Duration	$\sum_{i=1}^{\lVert\mathbb{D}\rVert} \mathbf{I}_k(\mathbb{D}_i)$	$[0, 16r)$
	Note Duration Difference	$\sum_{i=1}^{\lVert\mathbb{D}\rVert-1} \mathbf{I}_k(\mathbb{D}_{i+1} - \mathbb{D}_i + 16r)$	$[0, 32r)$
	Note Offset	$\sum_{i=1}^{\lVert\mathbb{O}\rVert} \mathbf{I}_k((\mathbb{O}_i + \mathbb{D}_i) \mod 16R)$	$[0, 16R)$
	Note Onset	$\sum_{i=1}^{\lVert\mathbb{O}\rVert} \mathbf{I}_k(\mathbb{O}_i \mod 4R)$	$[0, 4r)$,
	Note Onset Difference	$\sum_{i=1}^{\lVert\mathbb{O}\rVert-1} \mathbf{I}_k(\mathbb{O}_{i+1} - \mathbb{O}_i)$	$[0, 16r)$
	Pitch Interval	$\sum_{i=1}^{\lVert\mathbb{P}\rVert-1} \mathbf{I}_k(\mathbb{P}_{i+1} - \mathbb{P}_i + 128)$	$[0, 256)$
Polyphonic	Chord Duration $\star$	$\sum_{i=1}^{\lVert\mathbb{B}\rVert-1} \mathbf{I}(\lVert\mathbb{C}_i\rVert > 0)\mathbf{I}_k(\mathbb{B}_{i+1} - \mathbb{B}_i)$	$[0, 16\mathrm{r})$
	Chord Jaccard Distance	$\sum_{i=1}^{\lVert\mathbb{B}\rVert-2} \mathbf{I}_k\Big(\Big\lceil(d-1)\frac{\lVert\psi(\mathbb{C}_i) \cap \psi(\mathbb{C}_{i+1})\rVert}{\lVert\psi(\mathbb{C}_i) \cup \psi(\mathbb{C}_{i+1})\rVert} - 0.5\Big\rceil\Big)$	$[0, d)$
	Chord Onset	$\sum_{i=1}^{\lVert\mathbb{B}\rVert-1} \mathbf{I}_k\big(\sum_{j=1}^{\lVert\mathbb{C}_i\rVert}(1 \ll j)\mathbf{I}(\texttt{ons}(\mathbb{C}_i^j) = \mathbb{B}_i)\big)$	$[0, 352)$
	Chord Onset $\star$	$\sum_{i=1}^{\lVert\mathbb{B}\rVert-1} \mathbf{I}_k\big(\sum_{j=1}^{\lVert\mathbb{C}_i\rVert}(1 \ll j)\mathbf{I}(\texttt{ons}(\mathbb{C}_i^j) = \mathbb{B}_i)\big)(\mathbb{B}_{i+1} - \mathbb{B}_i)$	$[0, 352)$
	Chord Onset Difference	$\sum_{i=1}^{\lVert\mathbb{B}\rVert-1} \mathbf{I}_k(\mathbb{B}_{i+1} - \mathbb{B}_i + 128)$	$[0, 256)$
	Chord Onset PCD $\star$	$\sum_{i=1}^{\lVert\mathbb{B}\rVert-1} \mathbf{I}_k\big(\texttt{PCD}(\{\texttt{pitch}(x) \mod 12 : (x \in \mathbb{C}_i) \wedge (\texttt{ons}(x) = \mathbb{B}_i)\})\big)(\mathbb{B}_{i+1} - \mathbb{B}_i)$	$[0, 352)$
	Chord Outer Interval	$\sum_{i=1}^{\lVert\mathbb{B}\rVert-1} \mathbf{I}_k((\max(\psi(\mathbb{C}_i)) - \min(\psi(\mathbb{C}_i))) \mod 12)$	$[0, 12)$
	Chord PCD	$\sum_{i=1}^{\lVert\mathbb{B}\rVert-1} \mathbf{I}_k\big(\texttt{PCD}(\{\texttt{pitch}(x) \mod 12 : x \in \mathbb{C}_i\})\big)$	$[0, 352)$
	Chord PCD $\star$	$\sum_{i=1}^{\lVert\mathbb{B}\rVert-1} \mathbf{I}_k\big(\texttt{PCD}(\{\texttt{pitch}(x) \mod 12 : x \in \mathbb{C}_i\})\big)(\mathbb{B}_{i+1} - \mathbb{B}_i)$	$[0, 352)$
	Chord Size $\star$	$\sum_{i=1}^{\lVert\mathbb{B}\rVert-1} \mathbf{I}_k(\lVert\mathbb{C}_i\rVert)(\mathbb{B}_{i+1} - \mathbb{B}_i)$	$[0, 12)$
	Note Pitch	$\sum_{i=1}^{\lVert\mathbb{P}\rVert} \mathbf{I}_k(\mathbb{P}_i)$	$[0, 128)$

Table C.1: Formal definitions for the feature transformations used by monophonic, polyphonic and both models. $\star$ denotes feature transformations that are weighted by chord duration, using the term $(\mathbb{B}_{i+1} - \mathbb{B}_i)$. The domain $[a, b)$ sets the bounds of the categorical distribution. In our implementation, we set $d = 25$ and $r = 8$.

We use distinct pitch class sets (PCD) [3] to represent pitched material, which reduces the $2^{12} = 4096$ possible pitch class sets to 352 equivalence classes, grouping pitch class sets that are transpositionally equivalent. For example, the pitch class sets $\{0, 4, 7\}$ and $\{2, 5, 10\}$ are transpositionally equivalent, as both are major chords, the only difference being their root. $\texttt{PCD}(\cdot)$ is a function that accepts a pitch class set and returns an integer corresponding to the PCD. For more details on calculating the PCD, see the original paper [3].

We represent each musical excerpt $(\mathbb{M})$ by applying a non-empty set of feature transformations $\mathcal{F} = \{f_1, ..., f_d\}$, producing a set of categorical distributions $\mathcal{F}^{\mathbb{M}} = \{f_1^{\mathbb{M}} ..., f_d^{\mathbb{M}}\}$. A categorical distribution is a discrete probability distribution describing a random variable that has k possible distinct states. Concretely, $f_i^{\mathbb{M}} = [f_i(k) : a \leq k < b]$, where a and b are the upper and lower bounds of the domain respectively. The categorical distributions in $\mathcal{F}^{\mathbb{M}}$ are concatenated, resulting in a single vector representing $\mathbb{M}$, which we refer to as $\mathbf{v}^{\mathcal{F},\mathbb{M}}$. Table C.1 provides formal definitions for each of the feature transformations (f_i), and specifies the domain used to construct the corresponding categorical distribution $(f_i^{\mathbb{M}})$. Note that in some cases, the domain is dependant on the number of subdivisions per beat (r). Let $\mathbf{I}_k(\cdot)$ be a function that returns 1 if $\cdot = k$ and 0 otherwise.

C.2.2 Training

Given a prime $\mathbb{M}_{\text{prime}}$, and two possible continuations $(\mathbb{M}_a, \mathbb{M}_b)$, we train a Gradient Boosting Classifier [2] to predict whether $\mathbb{M}_a$ or $\mathbb{M}_b$ is the genuine continuation given $\mathbf{v}^{\mathbb{M}_{\text{prime}}} \oplus \mathbf{v}^{\mathbb{M}_a} \oplus \mathbf{v}^{\mathbb{M}_b}$

as input. Concretely, the classifier is trained to output a 0 if $\mathbb{M}_a$ is the genuine continuation and 1 otherwise. Notably, we were able to attain the same level of accuracy by training a Gradient Boosting Classifier to output 1 if the continuation $(\mathbb{M}_x)$ is genuine and 0 otherwise given $\mathbf{v}^{\mathbb{M}_{\text{prime}}} \oplus \mathbf{v}^{\mathbb{M}_x}$ as input.

The code was implemented in Python using the scikit-learn module [4]. Notably, a model can be trained on $10,000$ training examples in several minutes on an Intel Core i7-9700, which is much faster than training *BachProp*.

C.3 Acknowledgments

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

Bibliography

[1] Florian Colombo. *MIREX 2018: Generating and Discriminating Symbolic Music Continuations with BachProp*. https://www.music-ir.org/mirex/abstracts/2018/FC1.pdf. Accessed on August 19, 2019.

[2] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *Annals of Statistics* 29 (2000), pp. 1189–1232.

[3] **Jeff Ens** and Philippe Pasquier. "Quantifying Musical Style: Ranking Symbolic Music based on Similarity to a Style". In: *Proc. of the International Symposium on Music Information Retrieval*. 2019, pp. 870–877.

[4] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

Appendix D

Copyright Considerations for the MetaMIDI Dataset

When assembling a large dataset of MIDI files that have been scraped from publicly available internet sites, one must consider the implications with regards to copyrights. In order to better understand any legal issues related to assembling and distributing this dataset as researchers in Canada, we consulted Donald Taylor, a copyright officer at Simon Fraser University. In the following paragraph, Taylor's response is summarized.

According to Taylor, a MIDI file is considered to be a derived annotation of an original musical work, and thus, is treated the same as a machine generated transcription of a speech or lecture under copyright law. As a result, the rightsholder of the original musical work has the legal right to enforce copyright. Furthermore, if the rightsholder has not consented to the work being made freely available for download as a MIDI file, then these MIDI files would be considered a copyright infringing files. Consequently, using or distributing any of these files would be considered copyright infringement. Notably, in contrast to Canadian copyright law, the MetaMIDI dataset would be considered transformative use under the USA's fair use doctrine, since the dataset is not being used as a source of music for audio playback, but rather as a source of data to me mined for research purposes. Fortunately, there is a provision in Canada's Copyright Act, known as the "fair-dealing" exception, which allows for the use of copyright protected works to conduct research.

Ultimately, we were advised that it could reasonably be considered "fair-dealing" if we release the dataset under the following conditions.

1. Prospective users must demonstrate that they are using the dataset for research purposes related to data mining and machine learning. Concretely, this requires that the prospective user provides their name, institutional affiliation, institutional contact information, the name of their research project, and the location where the research is taking place.
2. Prospective users must acknowledge they will not further distribute the dataset.

In our best effort to acknowledge any copyrights for the MIDI files contained in the MetaMIDI dataset, we aggregated all the copyright metadata found in the MIDI files into a single text file, which is distributed as part of the dataset. The dataset is available on zenodo [1], with access restricted to users who meet conditions 1 and 2 as stated above.

[1]https://zenodo.org/record/5142664#.Y7xcBuzMKrM

Appendix E

StyleRank Appendix

The following is an appendix that complements the material presented in Chapter 5.

E.1 Features

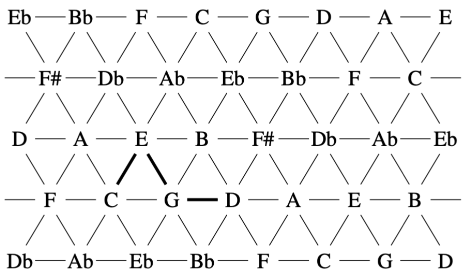
To begin, we provide additional explanations for some of the features that StyleRank uses. The function `sc` calculates the scale representation described in the text. Let $\mathbb{S}^{\texttt{maj}} = \{0, 2, 4, 5, 7, 9, 11\}$ and $\mathbb{S}^{\texttt{harm}} = \{0, 2, 3, 5, 7, 8, 11\}$, then $\mathbb{S}_i^{\texttt{maj}} = \{(s + i) \mod 12 : s \in \mathbb{S}^{\texttt{maj}}\}$ denotes the major scale with pitch class i as a root, and $\mathbb{S}_i^{\texttt{harm}} = \{(s+i) \mod 12 : s \in \mathbb{S}^{\texttt{harm}}\}$ denotes the harmonic minor scale with pitch class i as a root. The scale representation can be calculated using Equation E.1, where x is a pitch class set, and $\phi(\cdot)$ is a function that returns 1 if the predicate $\cdot$ is true, and 0 otherwise.

$$\texttt{sc}(x) = \Big(\sum_{i=1}^{12} \phi(x \subseteq \mathbb{S}_i^{\texttt{maj}}) \ll i\Big) + \Big(\sum_{i=1}^{12} \phi(x \subseteq \mathbb{S}_i^{\texttt{harm}}) \ll (12 + i)\Big) \tag{E.1}$$

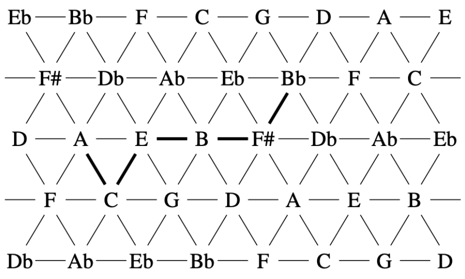
The difference between two pitches p_1 and p_2 can be represented in several ways: as the absolute difference between two pitches ($d_{\texttt{abs}}(p_1, p_2) = |p_1 - p_2|$); as an interval ($d_{\texttt{mod}}(p_1, p_2) = (p_1 - p_2) \mod 12$); and as an interval class. There are 6 interval classes: the unison ($d_{\texttt{mod}}(p_1, p_2) \in \{0\}$; the minor second and major seventh ($d_{\texttt{mod}}(p_1, p_2) \in \{1, 11\}$; the major second and minor seventh ($d_{\texttt{mod}}(p_1, p_2) \in \{2, 10\}$; the minor third and major sixth ($d_{\texttt{mod}}(p_1, p_2) \in \{3, 9\}$; the major third and minor sixth ($d_{\texttt{mod}}(p_1, p_2) \in \{4, 8\}$; the perfect fourth and fifth ($d_{\texttt{mod}}(p_1, p_2) \in \{5, 7\}$; and the tritone ($d_{\texttt{mod}}(p_1, p_2) \in \{6\}$. We calculate the interval class using Equation E.2,

$$\texttt{pcc}(p_1, p_2) = |((p_1 - p_2) \mod 12) - 6| \tag{E.2}$$

The `ChordTonnetz` feature is simply the length of the shortest path which passes through each of the pitch classes contained in a chord. For example, given a chord $\mathbb{C}$ and the corresponding pitch set $\mathbb{P} = \{48, 55, 60, 62, 64\}$, the shortest path passing though each pitch class in the set {C,D,E,G} is shown on the left side of Figure E.1. In this case, the length of the shortest path is 3. In general,



(a) The shortest path passing through C, D, E, and G.



(b) The shortest path passing through C, F, A, and Bb.

Figure E.1: Calculating the `ChordTonnetz` feature for different pitch class sets

harmonically simple chords will have shorter paths than harmonically complex chords. The shortest path for the pitch class set {C,F#,A,Bb} is shown on the right side of Figure E.1.

The `ChordTranVoiceMotion` (`CTVM`) feature denotes the type of voice motion between two successive chords. Given two successive pitch class sets $\mathbb{P}^t$ and $\mathbb{P}^{t+1}$ Equation E.3 is used to calculate the type of voice motion, where 0 is no change, 1 is oblique motion, 2 is parallel motion, and 3 is contrary motion.

$$\texttt{CTVM}(\mathbb{P}^t, \mathbb{P}^{t+1}) = \begin{cases} 0, & \text{if } (\min(\mathbb{P}^t) = \min(\mathbb{P}^{t+1})) \wedge (\max(\mathbb{P}^t) = \max(\mathbb{P}^{t+1})) \\ 1, & \text{if } (\min(\mathbb{P}^t) = \min(\mathbb{P}^{t+1})) \wedge (\max(\mathbb{P}^t) \neq \max(\mathbb{P}^{t+1})) \\ 1, & \text{if } (\min(\mathbb{P}^t) \neq \min(\mathbb{P}^{t+1})) \wedge (\max(\mathbb{P}^t) = \max(\mathbb{P}^{t+1})) \\ 2, & \text{if } (\min(\mathbb{P}^t) > \min(\mathbb{P}^{t+1})) \wedge (\max(\mathbb{P}^t) > \max(\mathbb{P}^{t+1})) \\ 2, & \text{if } (\min(\mathbb{P}^t) < \min(\mathbb{P}^{t+1})) \wedge (\max(\mathbb{P}^t) < \max(\mathbb{P}^{t+1})) \\ 3, & \text{otherwise} \end{cases} \tag{E.3}$$

Here, we briefly describe the method for calculating chord periodicity, which is used to calculate the `ChordDissonance` and `ChordTranDissonance` features. For more specific details please consult the original paper [32]. Given a set of pitches $\mathbb{P}_0 = \{0, 3, 9\}$, which corresponds to the frequency ratios $\mathbb{F}_0 = \{1/1, 6/5, 5/3\}$, the lowest common multiple of the denominators is calculated ($L_0 = \text{lcm}(1, 5, 3) = 15$). To calculate the smoothed periodicity, L_1 and L_2 are calculated using the shifted pitch sets $\mathbb{P}_1 = \{-3, 0, 6\}$ and $\mathbb{P}_2 = \{-9, -6, 0\}$. The L_i values are scaled, and the average of the three values is taken. `ChordTranDissonance` slighlty modifies the above procedure. Given two pitch sets $\mathbb{P}^t = \{0, 3, 9\}$ and $\mathbb{P}^{t+1} = \{0, 4, 7\}$, the L values are calculated for the pitch sets $\mathbb{P}_0 = \{0, 4, 7\}$, $\mathbb{P}_1 = \{-3, 1, 4\}$, $\mathbb{P}_2 = \{-9, -5, -2\}$. We construct these shifted pitch sets as follows, where $\mathbb{P}_i = \{p - \mathbb{P}^t_i : p \in \mathbb{P}^{t+1}\}$. The same scaling procedure is applied here, and the average of the L_i values is calculated. Since `ChordDissonance` and `ChordTranDissonance` must be turned into integers, we use the floor operator to turn a float into an integer. In order to have a larger number of categories, we do not apply the log transform to each L_i value.

E.2 Experiment 1 Expanded

In order to demonstrate that StyleRank is robust when the size of $\mathcal{G}$, and the number of styles in $\mathcal{G}$ are varied, we modify Experiment 1. Given $2+k$ styles $S^i = \{s_1^i, ..., s_{m_j}^i\}$, where $m_1 = 2n$, $m_j = n$ for $j > 1$, let $\mathcal{C} = \{s_i^1 : n < i \leq 2n\}$, $\mathcal{G}_A = \{s_i^1 : 0 \leq i < n\}$, $\mathcal{G}_B = \{s_i^2 : 0 \leq i < n\}$, $\mathcal{G} = \{s_i^j : (0 \leq i < n) \wedge (1 \leq j \leq 2+k)\}$. We train a random forest and compare two distributions $x = [S_g^{\mathcal{G},\mathcal{C},\mathcal{F}} : g \in \mathcal{G}_A]$ and $y = [S_g^{\mathcal{G},\mathcal{C},\mathcal{F}} : g \in \mathcal{G}_B]$. When $k = 0$, the process is identical to Experiment 1. The results for $0 \leq k < 4$ are shown in Table E.1, with the top scores bolded for each combination of corpus size (n) and style type (`genre`, `composer`).

On a whole, the performance does not decrease as k increases, evidenced by the fact that $k > 0$ models had better scores than $k = 0$ in many cases. In the most extreme case, with $k = 3$ and $n = 100$, the size of $\mathcal{C}$ and $\mathcal{G}$ vary significantly, where $||\mathcal{C}|| = 100$ and $||\mathcal{G}|| = 500$. We believe this provides compelling evidence that StyleRank is robust against discrepancies in size between $\mathcal{C}$ and $\mathcal{G}$, and variations to the number of styles in $\mathcal{G}$.

			StyleRank				Cosine				Manhattan				Euclidean			
	k	size	μ	Sig	FDR	Bon	μ	Sig	FDR	Bon	μ	Sig	FDR	Bon	μ	Sig	FDR	Bon
Genre	0	10	**0.81**	**0.379**	0.0	0.0	0.68	0.193	0.0	0.0	0.686	0.2	0.0	0.0	0.645	0.176	0.0	0.0
	1	10	0.802	0.355	0.0	0.0	0.689	0.183	0.0	0.0	0.672	0.194	0.0	0.0	0.633	0.164	0.0	0.0
	2	10	0.764	0.332	0.0	0.0	0.695	0.212	0.0	0.0	0.68	0.189	0.0	0.0	0.647	0.155	0.0	0.0
	3	10	0.744	0.314	0.0	0.0	0.713	0.209	0.0	0.0	0.687	0.209	0.0	0.0	0.651	0.168	0.0	0.0
	0	25	**0.867**	0.578	**0.376**	**0.198**	0.729	0.348	0.084	0.038	0.74	0.374	0.053	0.021	0.691	0.298	0.06	0.022
	1	25	0.858	0.567	0.339	0.181	0.714	0.343	0.084	0.034	0.752	0.361	0.027	0.012	0.674	0.276	0.037	0.017
	2	25	0.841	**0.592**	0.357	0.197	0.73	0.359	0.068	0.019	0.743	0.363	0.048	0.016	0.71	0.289	0.038	0.017
	3	25	0.846	0.581	0.352	0.179	0.736	0.338	0.068	0.024	0.722	0.363	0.041	0.011	0.696	0.286	0.033	0.009
	0	50	0.88	0.715	0.59	0.432	0.776	0.484	0.266	0.126	0.747	0.489	0.253	0.088	0.714	0.344	0.158	0.082
	1	50	0.897	**0.748**	**0.621**	**0.458**	0.731	0.442	0.236	0.126	0.773	0.488	0.258	0.088	0.744	0.375	0.19	0.117
	2	50	**0.901**	0.739	0.608	0.437	0.755	0.449	0.239	0.118	0.74	0.492	0.287	0.101	0.728	0.366	0.18	0.109
	3	50	0.892	0.732	0.577	0.409	0.772	0.451	0.22	0.105	0.752	0.474	0.273	0.097	0.738	0.372	0.172	0.103
	0	100	0.927	0.847	0.774	0.671	0.766	0.555	0.406	0.265	0.755	0.566	0.44	0.284	0.785	0.462	0.269	0.178
	1	100	**0.941**	0.855	0.794	0.679	0.768	0.541	0.406	0.279	0.741	0.541	0.416	0.291	0.783	0.439	0.265	0.189
	2	100	0.93	0.835	0.775	0.668	0.77	0.562	0.395	0.26	0.753	0.547	0.446	0.308	0.777	0.457	0.264	0.192
	3	100	0.937	**0.865**	**0.802**	**0.697**	0.758	0.551	0.4	0.277	0.732	0.564	0.458	0.302	0.785	0.468	0.274	0.199
Composer	0	10	0.963	**0.86**	**0.725**	0.0	0.837	0.624	0.381	0.0	0.879	0.662	0.413	0.0	0.827	0.565	0.28	0.0
	1	10	**0.971**	0.849	0.704	0.0	0.85	0.626	0.401	0.0	0.862	0.628	0.392	0.0	0.812	0.568	0.292	0.0
	2	10	0.96	0.838	0.646	0.0	0.873	0.633	0.392	0.0	0.843	0.631	0.369	0.0	0.83	0.579	0.314	0.0
	3	10	0.954	0.818	0.657	0.0	0.857	0.635	0.388	0.0	0.876	0.663	0.437	0.0	0.835	0.576	0.288	0.0
	0	25	0.951	**0.888**	0.807	0.609	0.808	0.583	0.422	0.24	0.793	0.578	0.415	0.244	0.729	0.532	0.363	0.226
	1	25	0.946	0.879	**0.812**	**0.613**	0.773	0.561	0.403	0.246	0.818	0.627	0.461	0.276	0.738	0.533	0.36	0.196
	2	25	0.941	0.88	0.808	**0.613**	0.782	0.58	0.422	0.248	0.797	0.601	0.424	0.221	0.746	0.546	0.38	0.219
	3	25	**0.952**	0.881	0.796	0.61	0.802	0.57	0.422	0.246	0.798	0.63	0.431	0.247	0.764	0.568	0.377	0.219
	0	50	0.926	0.905	0.873	0.78	0.705	0.559	0.454	0.333	0.751	0.599	0.468	0.34	0.717	0.565	0.428	0.3
	1	50	**0.943**	**0.917**	0.885	0.785	0.722	0.567	0.444	0.308	0.697	0.54	0.435	0.305	0.686	0.531	0.369	0.262
	2	50	0.936	0.902	0.869	0.784	0.732	0.574	0.465	0.326	0.743	0.578	0.453	0.318	0.693	0.549	0.437	0.312
	3	50	0.942	**0.917**	**0.892**	**0.787**	0.714	0.557	0.448	0.316	0.721	0.571	0.454	0.325	0.676	0.54	0.421	0.306
	0	100	**1.0**	0.986	0.973	**0.951**	0.713	0.636	0.59	0.515	0.723	0.633	0.568	0.486	0.715	0.626	0.571	0.504
	1	100	**1.0**	**0.988**	**0.978**	0.941	0.745	0.662	0.604	0.539	0.724	0.636	0.572	0.492	0.712	0.62	0.552	0.489
	2	100	0.997	0.986	0.97	0.929	0.709	0.623	0.577	0.52	0.731	0.645	0.579	0.5	0.7	0.63	0.572	0.504
	3	100	0.994	0.975	0.961	0.918	0.71	0.631	0.585	0.505	0.753	0.665	0.606	0.523	0.689	0.61	0.551	0.494

Table E.1: The normalized frequency over 1000 trials where $\bar{x} > \bar{y}$ (μ), $p^{\bar{x}>\bar{y}} < 0.05$ (Sig), $p^{\bar{x}>\bar{y}}$ is significant after applying the FDR correction (FDR), and $p^{\bar{x}>\bar{y}}$ is significant after applying the Bonferonni correction (Bon). Size denotes the size of the corpus $||\mathcal{C}|| = ||\mathcal{G}_k|| = ||\mathcal{G}_B||$.

E.3 Experiment 1 Data

The number of pieces belonging to each genre and composer after duplicates have been removed are shown in Table E.2 and E.3 respectively. Only the composers with more than $2n$ pieces are selected for comparison. As a result, there are only 8 composers to compare when $n = 100$.

Genre	Count
Middle Romantic	447
Post Romantic	515
Late Romantic	577
Early Romantic	777
Late Classical Early Romantic	1100
Late Baroque	2615

Table E.2: The number of pieces in each genre.

Composer	Count	Composer	Count	Composer	Count
Johann Sebastian Bach	1081	Johann Friedrich Burgmüller	56	Sir Arthur Sullivan	30
Wolfgang Amadeus Mozart	813	Henry Purcell	54	Michael Maier	30
Domenico Scarlatti	551	Georg Philipp Telemann	54	Thomas Morley	29
George Frideric Handel	482	Sergey Vasilyevich Rachmaninov	52	Johann Adolf Hasse	28
Ludwig Van Beethoven	384	Maurice Ravel	52	Bedrich Smetana	27
Franz Liszt	309	Domenico Zipoli	48	Stephen Heller	26
(franz) Joseph Haydn	308	Muzio Clementi	45	Nikolay Rimsky-korsakov	26
Antonio Vivaldi	212	Igor Stravinsky	44	Jean-philippe Rameau	26
Frédéric François Chopin	197	Carl Maria Von Weber	44	Jean-baptiste Lully	26
Franz Peter Schubert	197	Niccolò Paganini	42	Giovanni Pierluigi Da Palestrina	26
Johannes Brahms	184	Jose Mauricio Nunes García	39	Scott Joplin	25
Felix Mendelssohn-bartholdy	141	Erik Satie	39	Leopold Godowsky	25
Johann Nepomuk Hummel	135	Charles-valentin Alkan	39	Gustav Mahler	25
Pyotr Il'yich Tchaikovsky	125	Silvius Leopold Weiss	37	Karl Joachim Andersen	24
Antonín (leopold) Dvořák	125	John Philip Sousa	37	Jacques Offenbach	24
Robert Alexander Schumann	117	John Dowland	37	Anton Bruckner	24
Achille-claude Debussy	103	(wilhelm) Richard Wagner	37	Sir Edward Elgar	23
William Byrd	92	Mauro Giuliani	35	Giovanni Battista Pergolesi	23
Carl Czerny	81	Marin Marais	35	Lorenzo Perosi	22
Camille Saint-saëns	81	Giovanni Battista Sammartini	35	Jean-baptiste Lemire	22
Alexander Scriabin	77	Charles Gounod	35	Giacomo Puccini	22
Isaac Albéniz	68	Béla Bartók	35	Richard Walthew	21
Fernando Sor	66	Johann Pachelbel	34	Ferdinando Carulli	21
Gabriel Fauré	64	Georges Bizet	34	Adriano Banchieri	21
Edvard Grieg	62	Modest Petrovich Mussorgsky	33	Gaetano Donizetti	20

Table E.3: The number of pieces per composer.

E.4 Experiment 2 Data

In Table E.6, we provide the raw frequency counts for each piece, from which our ground truth ranking is constructed. We also calculated the percentage of pairwise comparisons that were identical for different participant levels (Novice, Intermediate, Advanced and Expert), shown in Table 8. The ranking constructed from Novice data is the most dissimilar from the other three levels. For varying levels of α, we show the number of significant comparisons for each level in Table E.5.

	Novice	Intermediate	Advanced	Expert
Novice	1.0	0.765	0.765	0.747
Intermediate	0.765	1.0	0.904	0.873
Advanced	0.765	0.904	1.0	0.870
Expert	0.747	0.873	0.870	1.0

Table E.4: Percentage of identical pairwise comparisons, based on data from the Bachbot experiment [1].

	$\alpha = 5.0$	$\alpha = 0.5$	$\alpha = 0.05$	$\alpha = 0.005$
Novice	630	432	174	82
Intermediate	630	554	429	351
Advanced	630	533	385	304
Expert	630	484	262	150

Table E.5: Number of comparisons below significance level, based on data from the Bachbot experiment [1].

	Novice		Intermediate		Advanced		Expert	
	N^{corr}	N^{miss}	N^{corr}	N^{miss}	N^{corr}	N^{miss}	N^{corr}	N^{miss}
BWV-310-mask-Alto-Tenor	113	53	308	70	142	26	53	3
BWV-378-mask-Alto-Tenor-Bass	121	57	267	66	146	31	73	6
BWV-11.6-mask-Alto-Tenor	107	71	234	88	124	38	49	6
BWV-419-mask-Soprano	129	39	289	69	142	28	68	9
BWV-430-mask-Alto-Tenor	116	60	290	63	142	17	61	9
BWV-411-mask-Alto-Tenor	107	73	256	96	128	28	53	8
BWV-121.6-mask-Soprano	108	68	254	87	159	39	63	11
BWV-276-mask-Alto-Tenor-Bass	120	65	245	113	146	34	48	10
BWV-372-mask-Alto-Tenor-Bass	97	59	269	78	130	35	52	12
BWV-127.5-mask-Alto-Tenor	74	72	233	104	125	42	59	14
BWV-381-mask-Alto-Tenor	112	60	286	74	143	24	45	11
BWV-166.6-mask-Alto-Tenor	94	69	296	77	151	32	53	14
BWV-65.2-mask-Alto-Tenor	91	57	205	106	130	49	57	16
BWV-268-mask-Alto	92	59	247	108	126	38	54	16
out-28	124	100	305	186	139	62	60	19
BWV-154.3-mask-Alto-Tenor-Bass	97	68	219	138	125	50	52	17
out-45	127	108	283	207	154	80	66	23
out-59	122	103	280	174	160	72	53	19
BWV-168.6-mask-Alto-Tenor-Bass	89	74	221	103	113	36	38	14
BWV-425-mask-Alto-Tenor-Bass	96	57	266	83	127	32	56	21
out-54	116	100	311	158	151	66	68	26
out-56	102	113	289	167	165	44	52	21
out-19	138	92	264	171	164	65	66	27
out-20	108	106	253	175	136	93	47	24
BWV-438-mask-Alto	107	75	204	152	111	50	41	22
BWV-270-mask-Alto-Tenor-Bass	89	77	213	153	88	78	33	18
out-33	136	96	290	191	131	99	57	32
BWV-114.7-mask-Tenor	79	74	198	147	104	64	47	27
out-60	93	129	224	242	118	97	55	37
BWV-248.5-mask-Alto-Tenor-Bass	90	73	177	175	91	79	36	25
out-63	99	116	233	261	113	108	51	43
BWV-102.7-mask-Tenor	88	75	191	154	89	60	21	19
BWV-27.6-mask-Bass	91	71	186	174	89	77	34	31
BWV-293-mask-Bass	77	97	166	160	78	87	24	24
out-10	131	93	222	208	109	129	39	51
out-52	125	81	235	243	84	143	36	50
total	3805	2840	8909	5021	4573	2132	1820	735

Table E.6: Raw count data from the BachBot experiment [1], where N^{miss} is the number of times a generated sample was mistakenly classified as a Bach chorale and N^{corr} is the number of times it was correctly identified as computer generated.

Bibliography

[1] Feynman Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. “Automatic Stylistic Composition of Bach Chorales with Deep LSTM.” In: *Proceedings of the International Symposium on Music Information Retrieval*. 2017, pp. 449–456.

Appendix F

Examples

In this Appendix, we present examples generated using the Multi-Track Music Machine (MMM), which is described in Chapter 7. We provide examples of unconditioned generation, track-infilling, attribute control and bar-infilling.

F.1 Unconditioned Generation

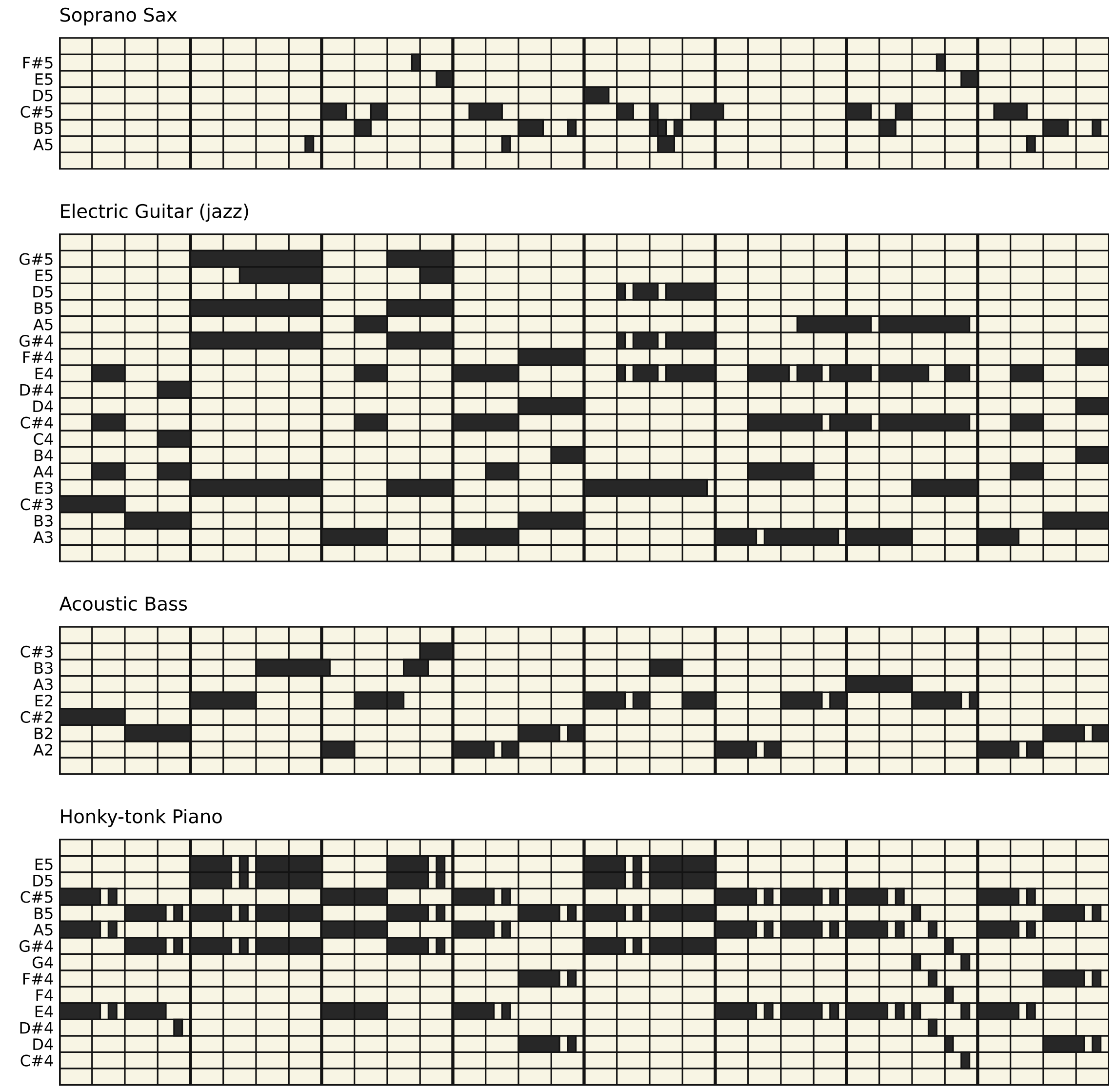


Figure F.1: An example of unconditioned multi-track generation using MMM.

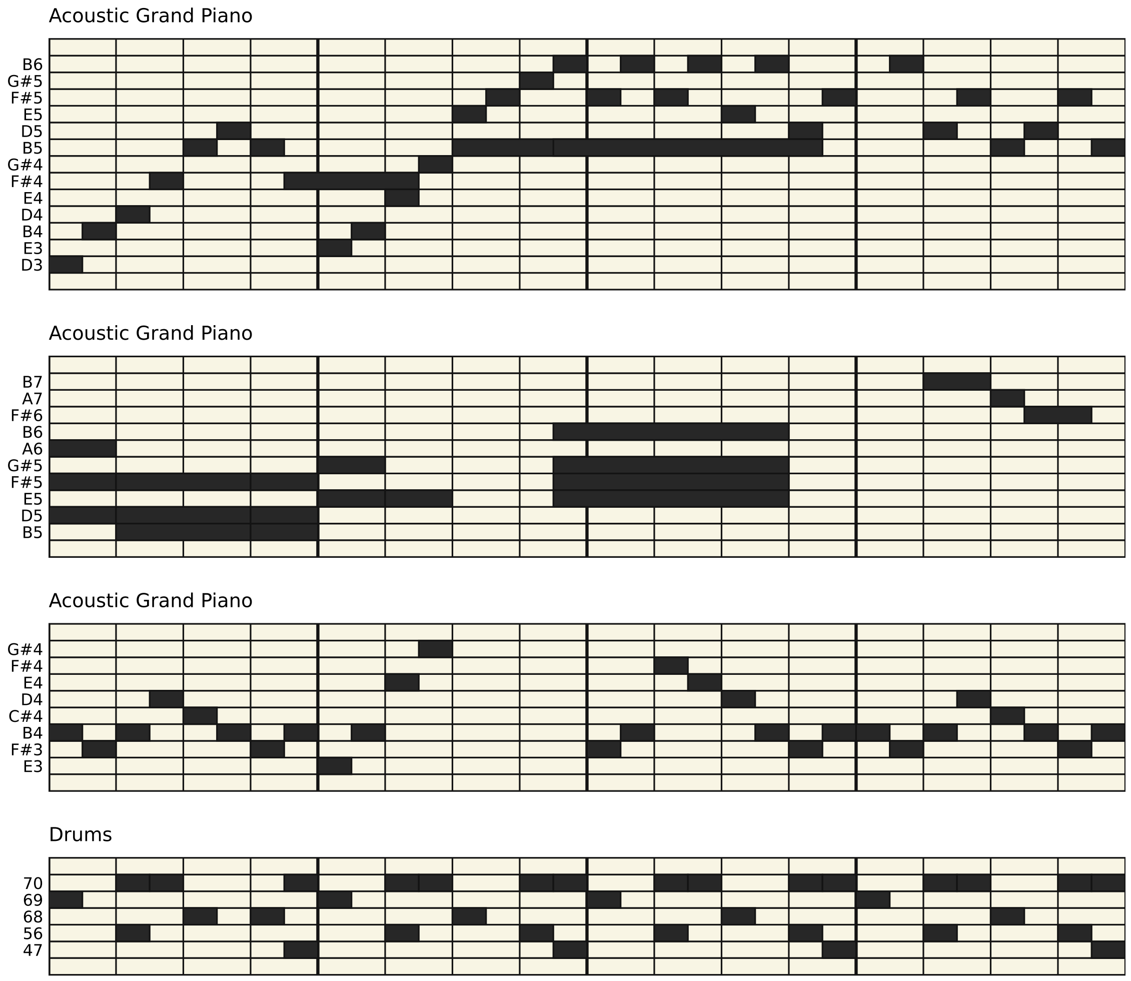


Figure F.2: An example of unconditioned multi-track generation using MMM.

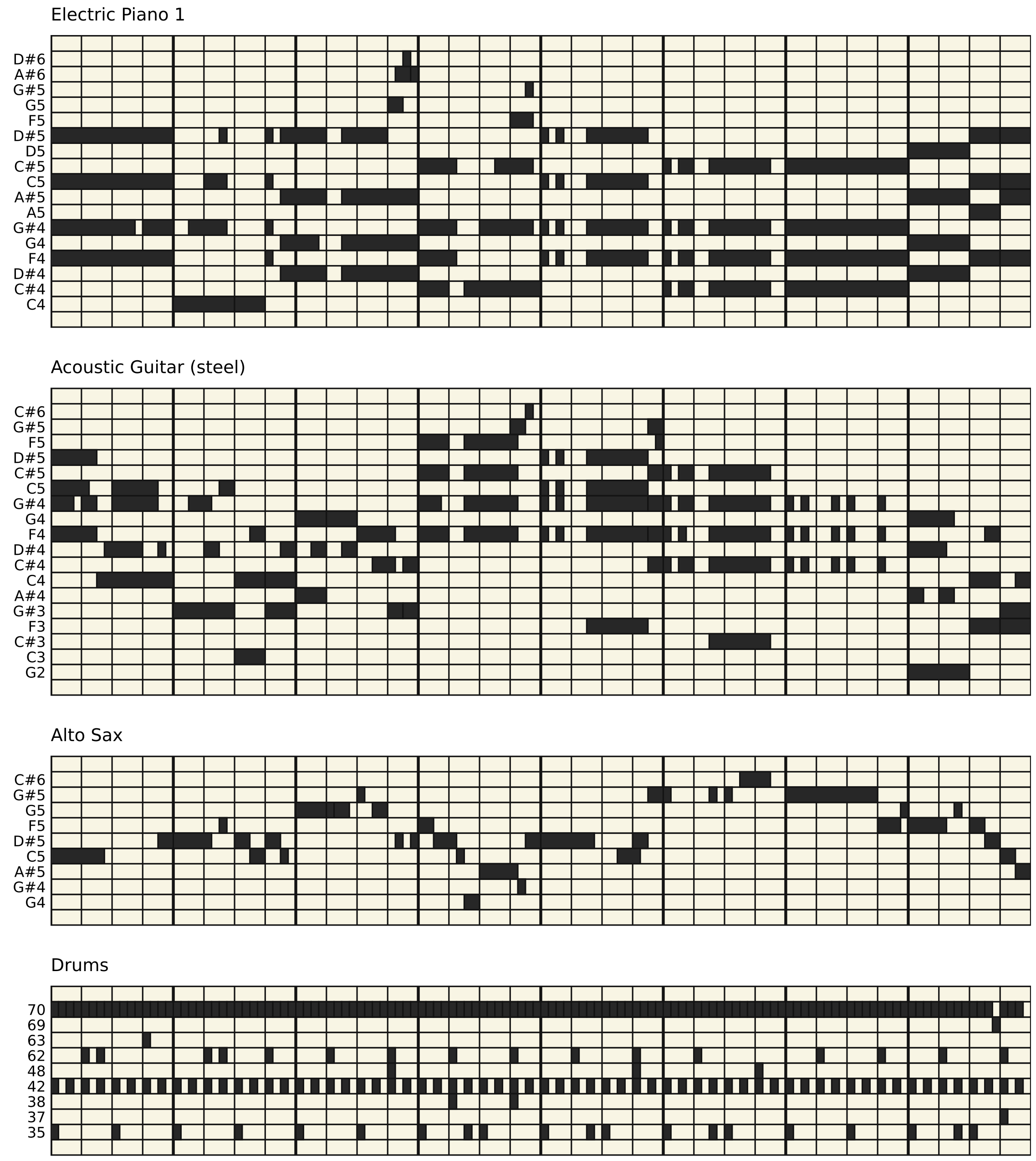


Figure F.3: An example of unconditioned multi-track generation using MMM.

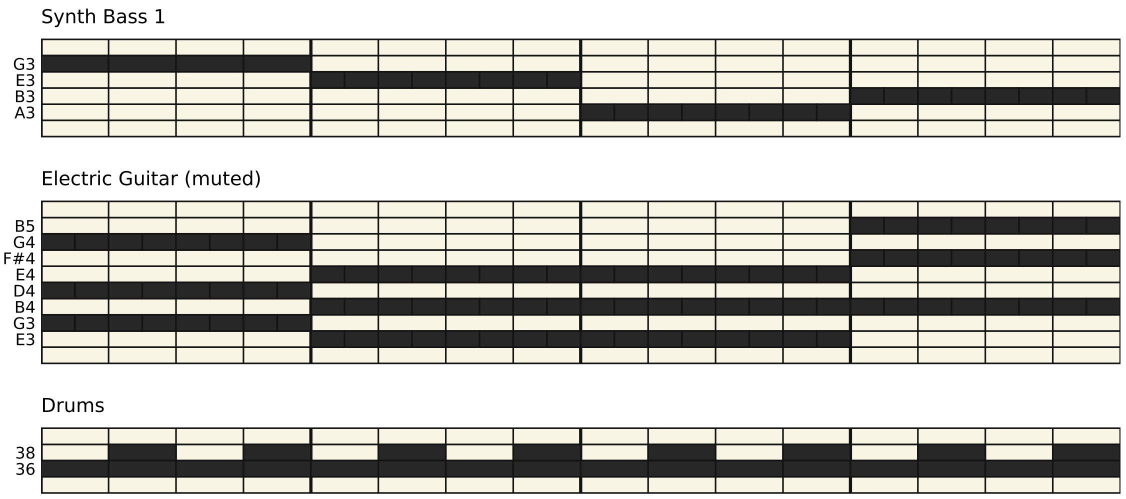


Figure F.4: An example of unconditioned multi-track generation using MMM.

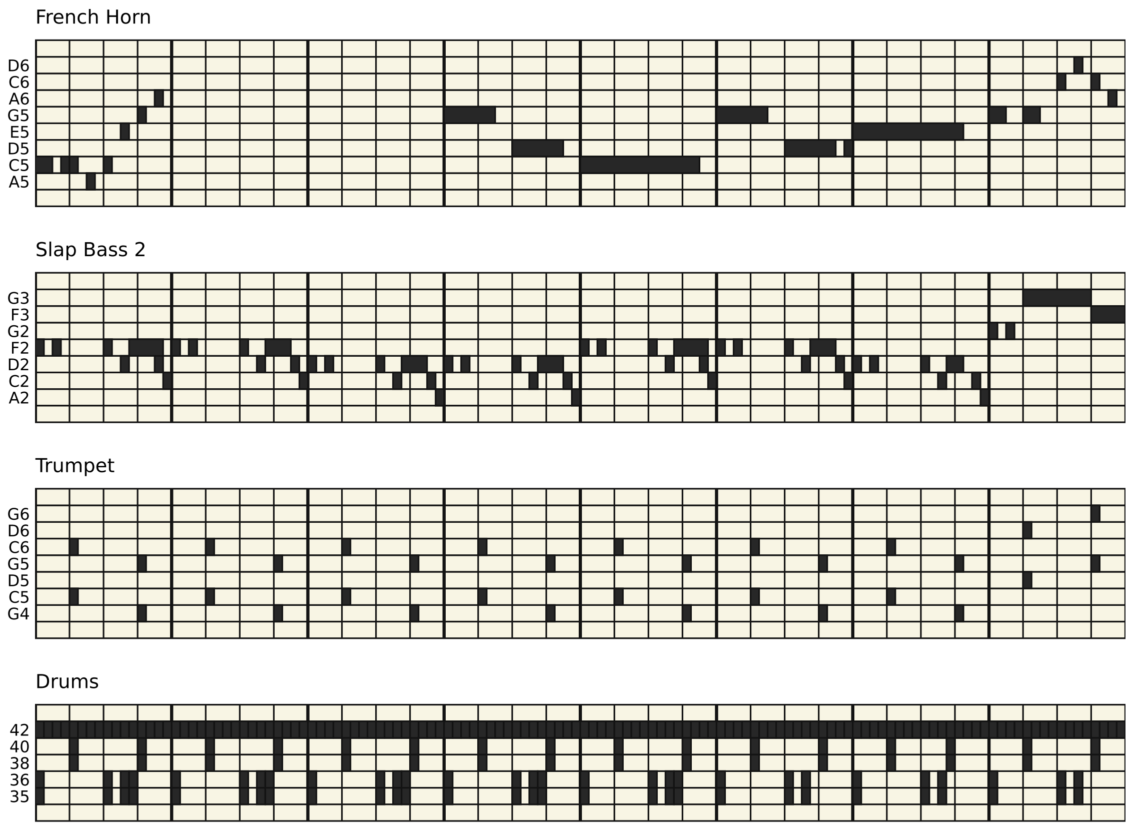


Figure F.5: An example of unconditioned multi-track generation using MMM.

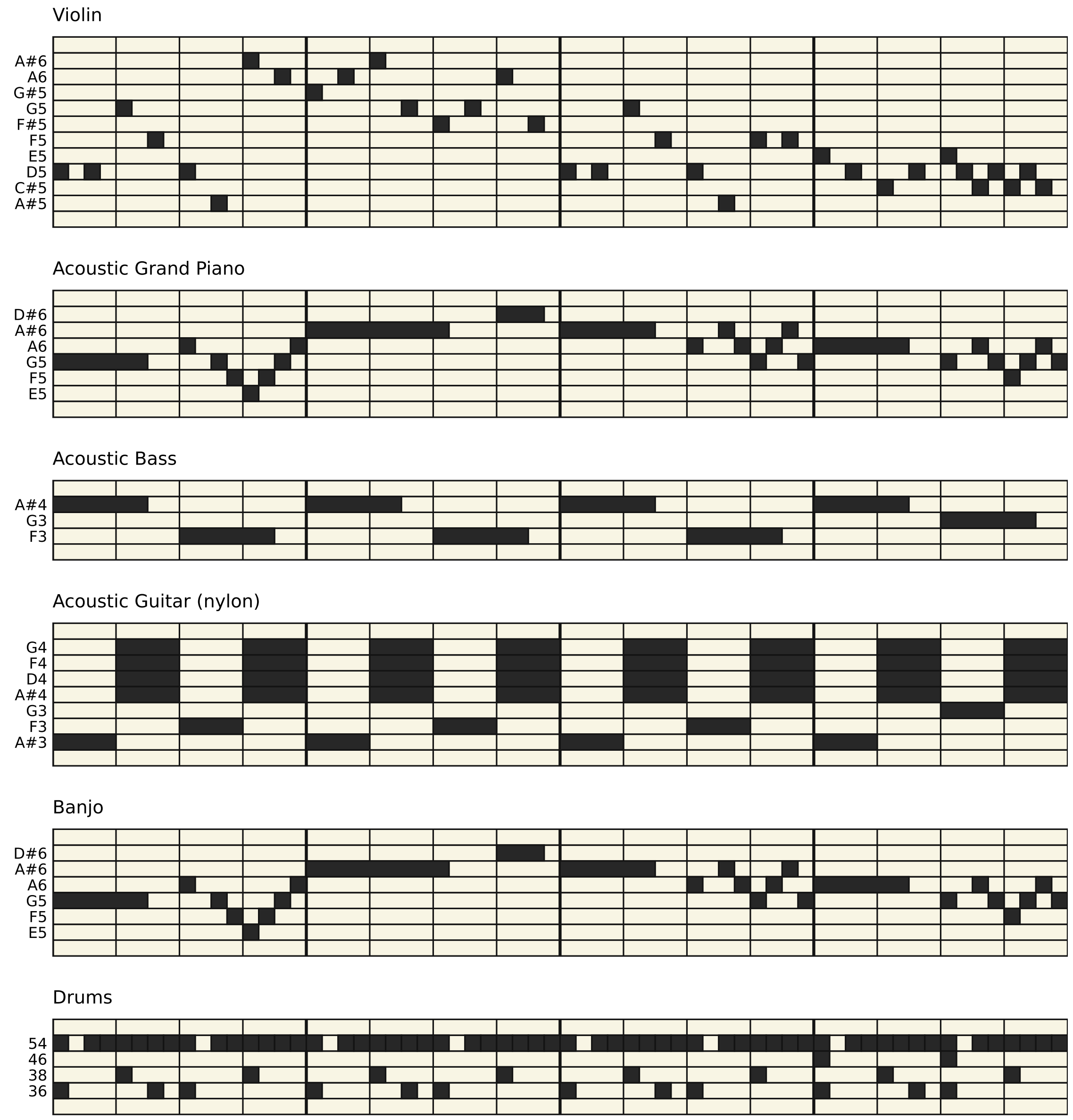


Figure F.6: An example of unconditioned multi-track generation using MMM.

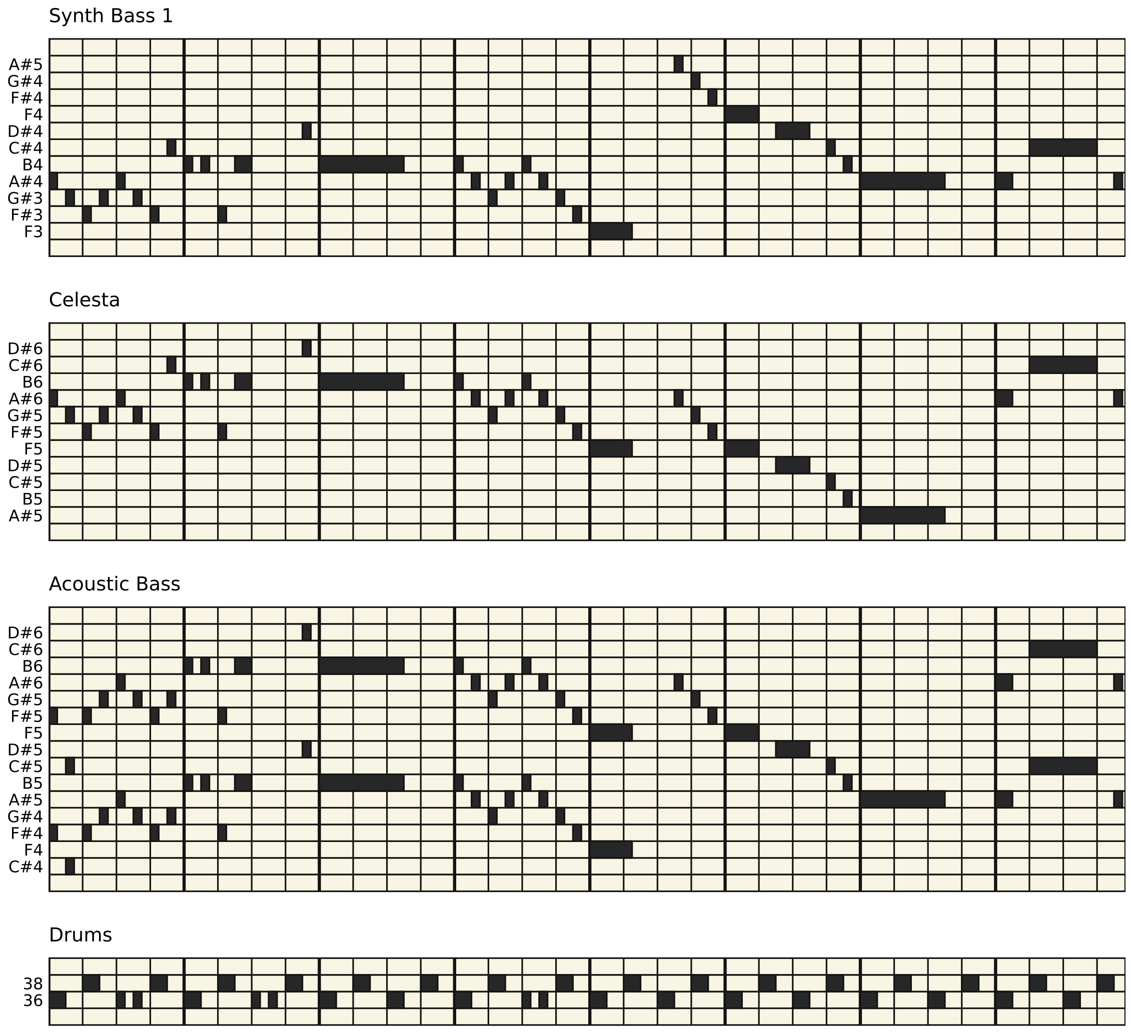


Figure F.7: An example of unconditioned multi-track generation using MMM.

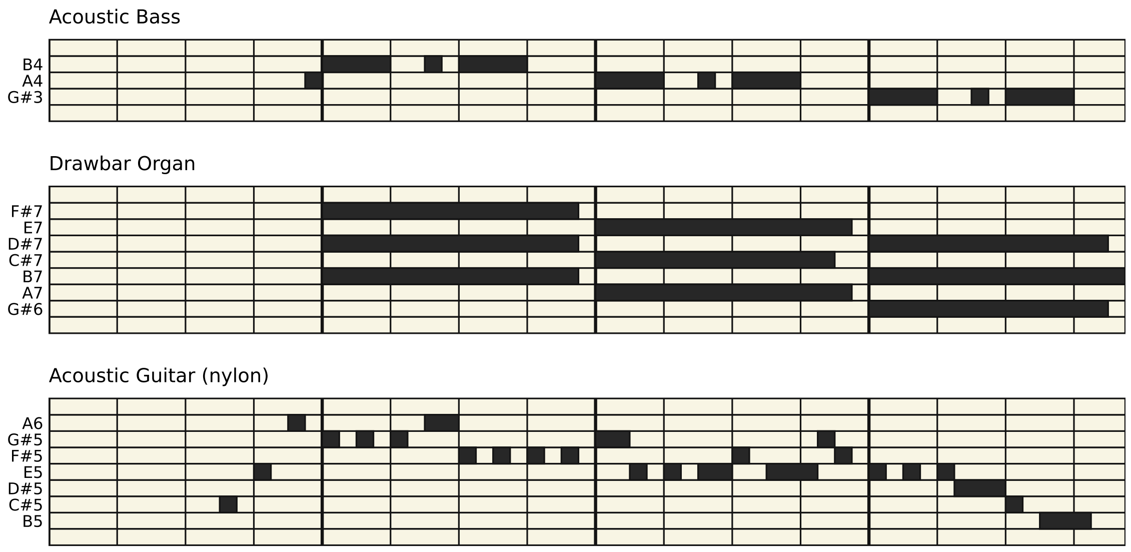


Figure F.8: An example of unconditioned multi-track generation using MMM.

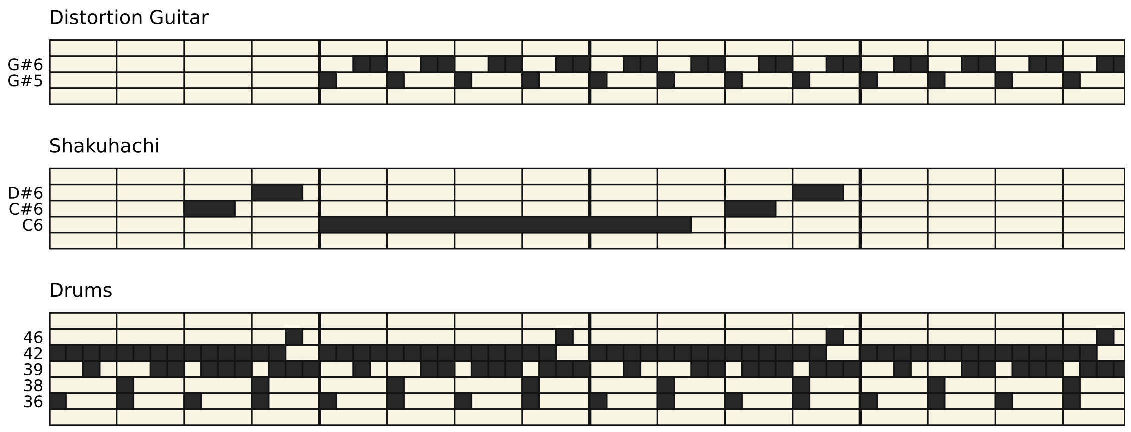


Figure F.9: An example of unconditioned multi-track generation using MMM.

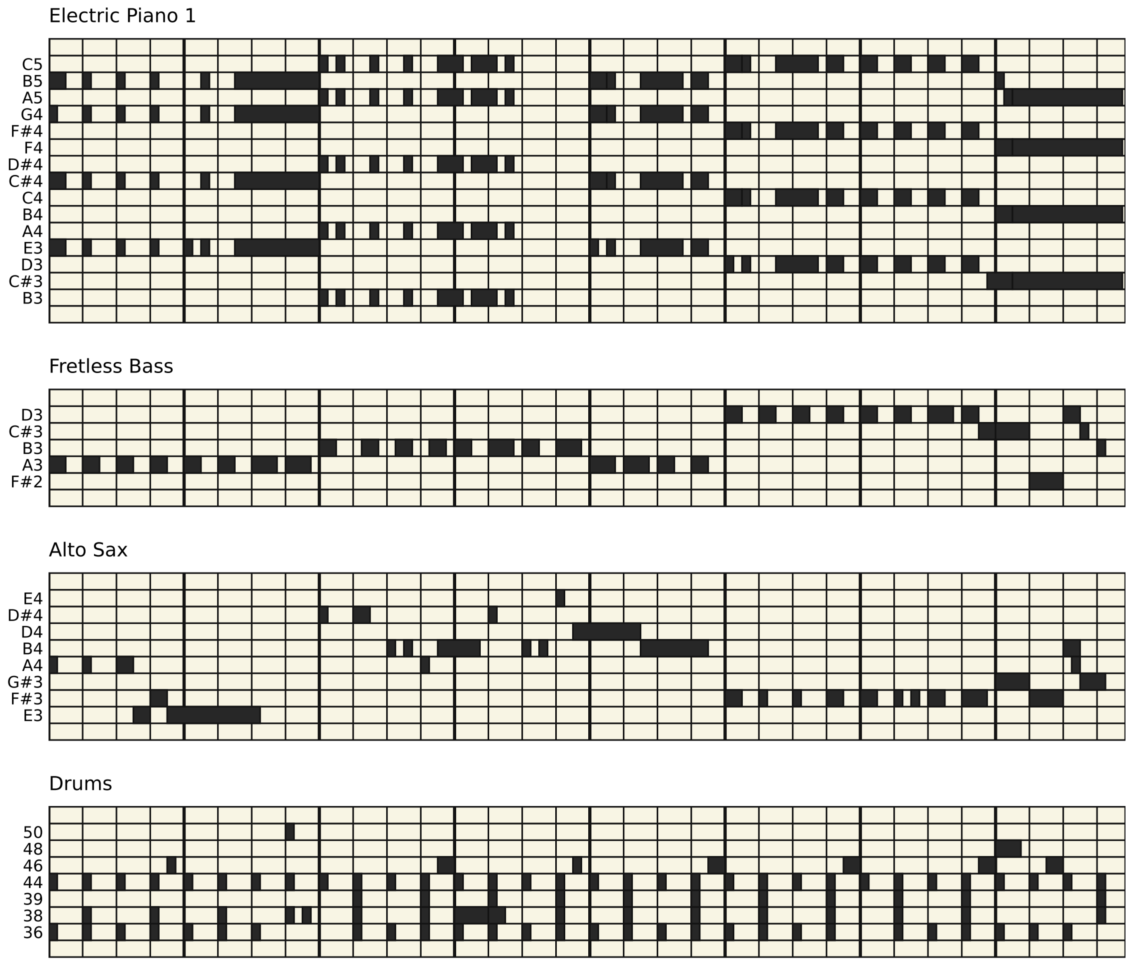


Figure F.10: An example of unconditioned multi-track generation using MMM.

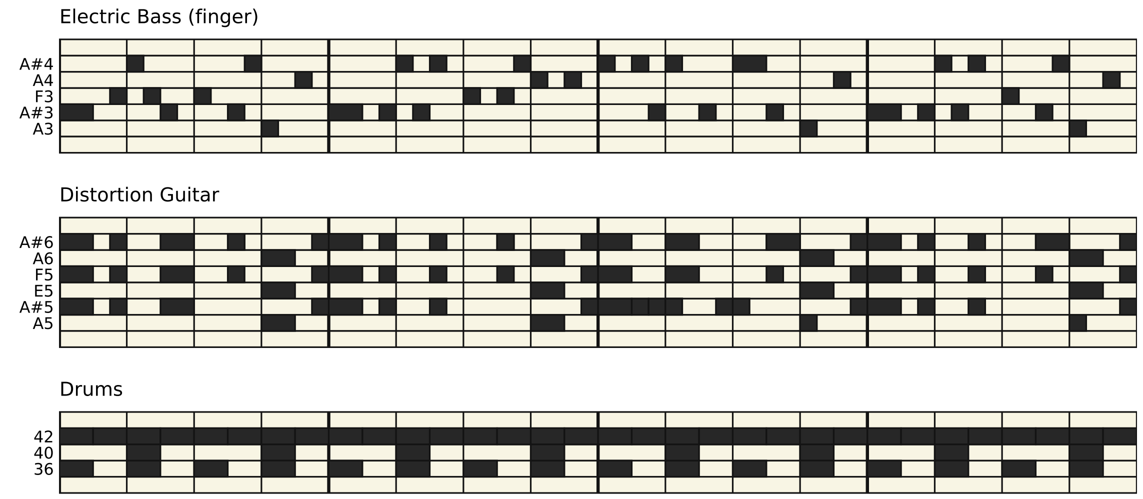


Figure F.11: An example of unconditioned multi-track generation using MMM.

F.2 Track Infilling

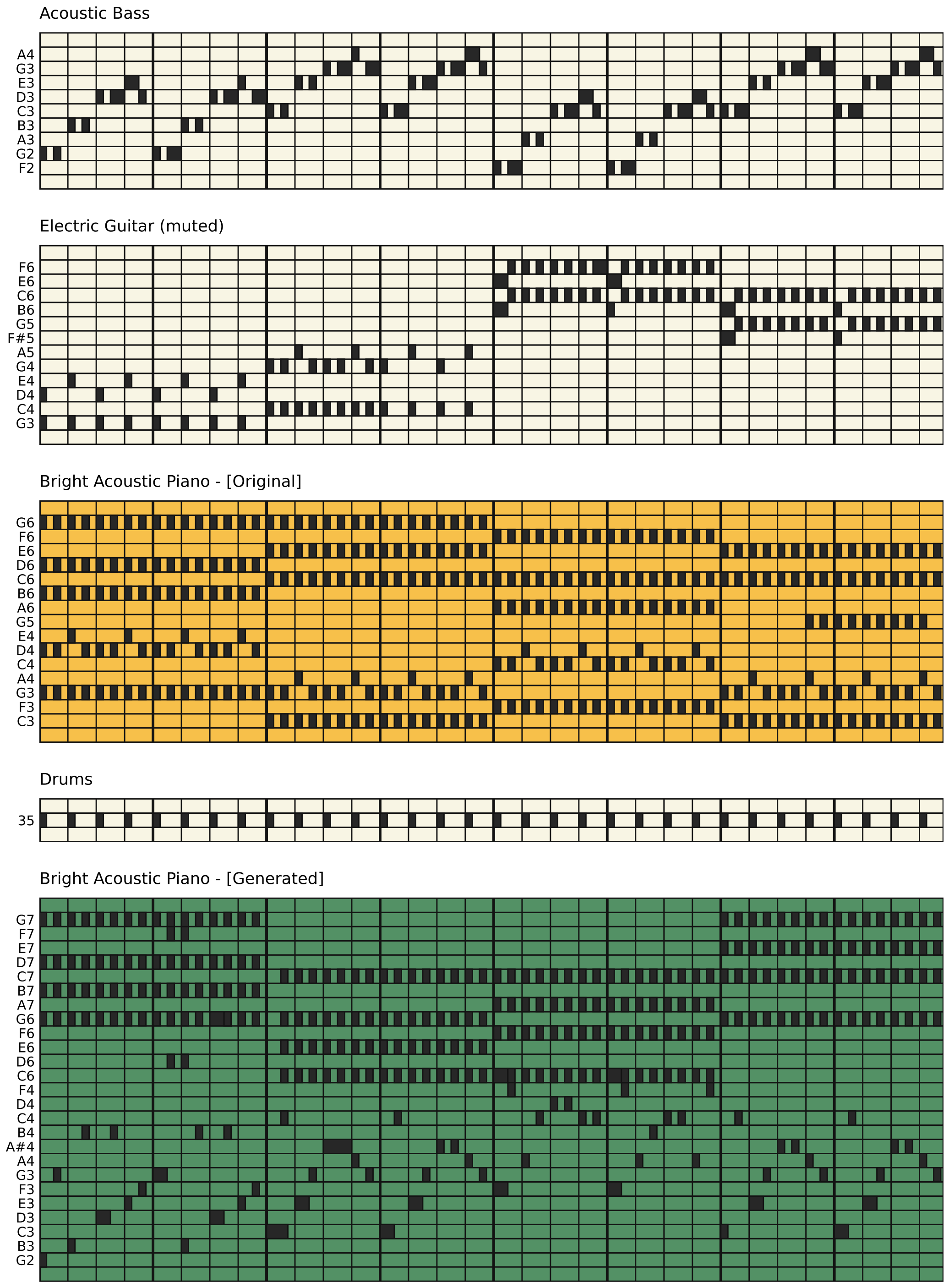


Figure F.12: An example of track infilling using MMM. The generation of each green track is conditioned on the beige tracks, replacing the original track(s) (shown in yellow).

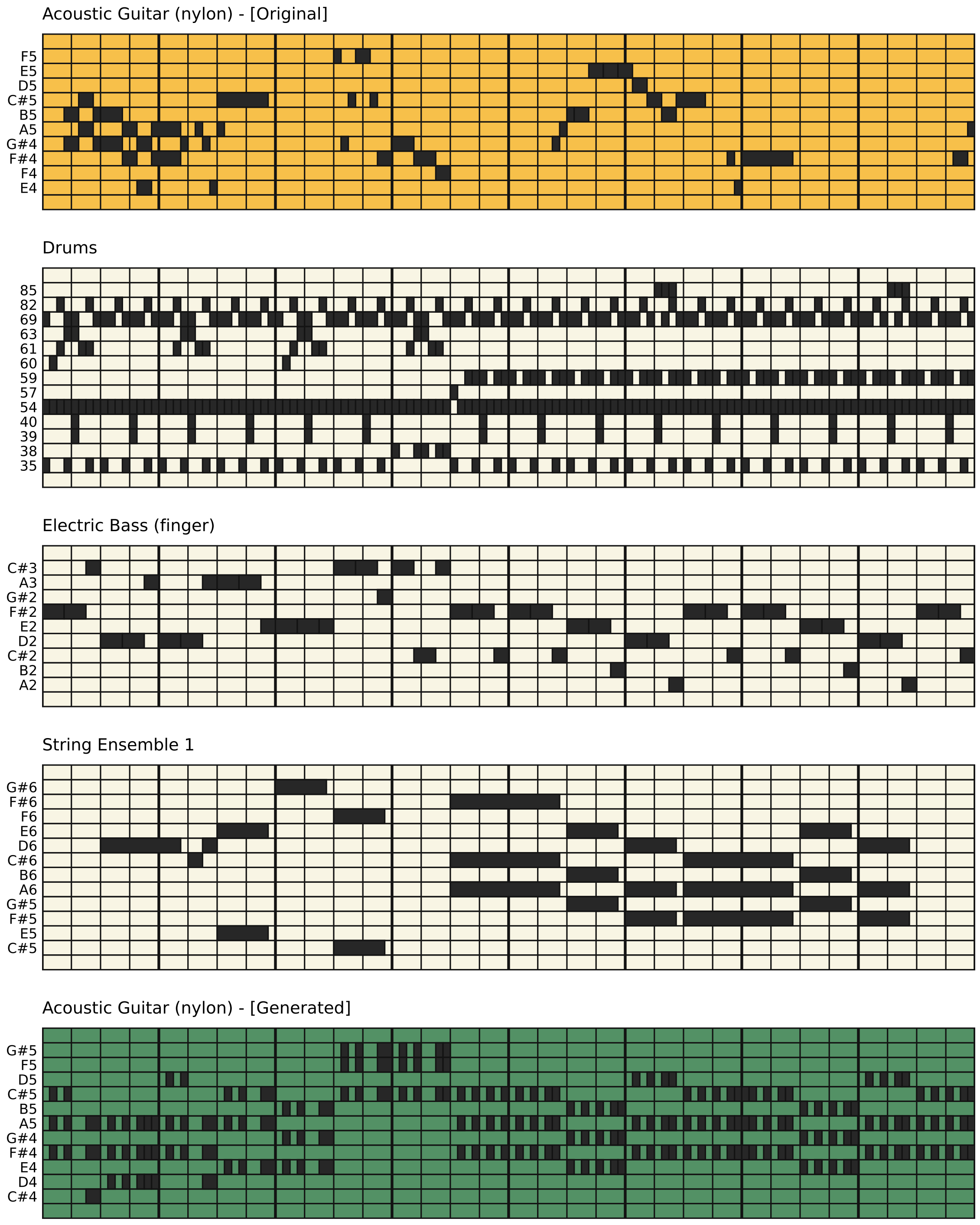


Figure F.13: An example of track infilling using MMM. The generation of each green track is conditioned on the beige tracks, replacing the original track(s) (shown in yellow).

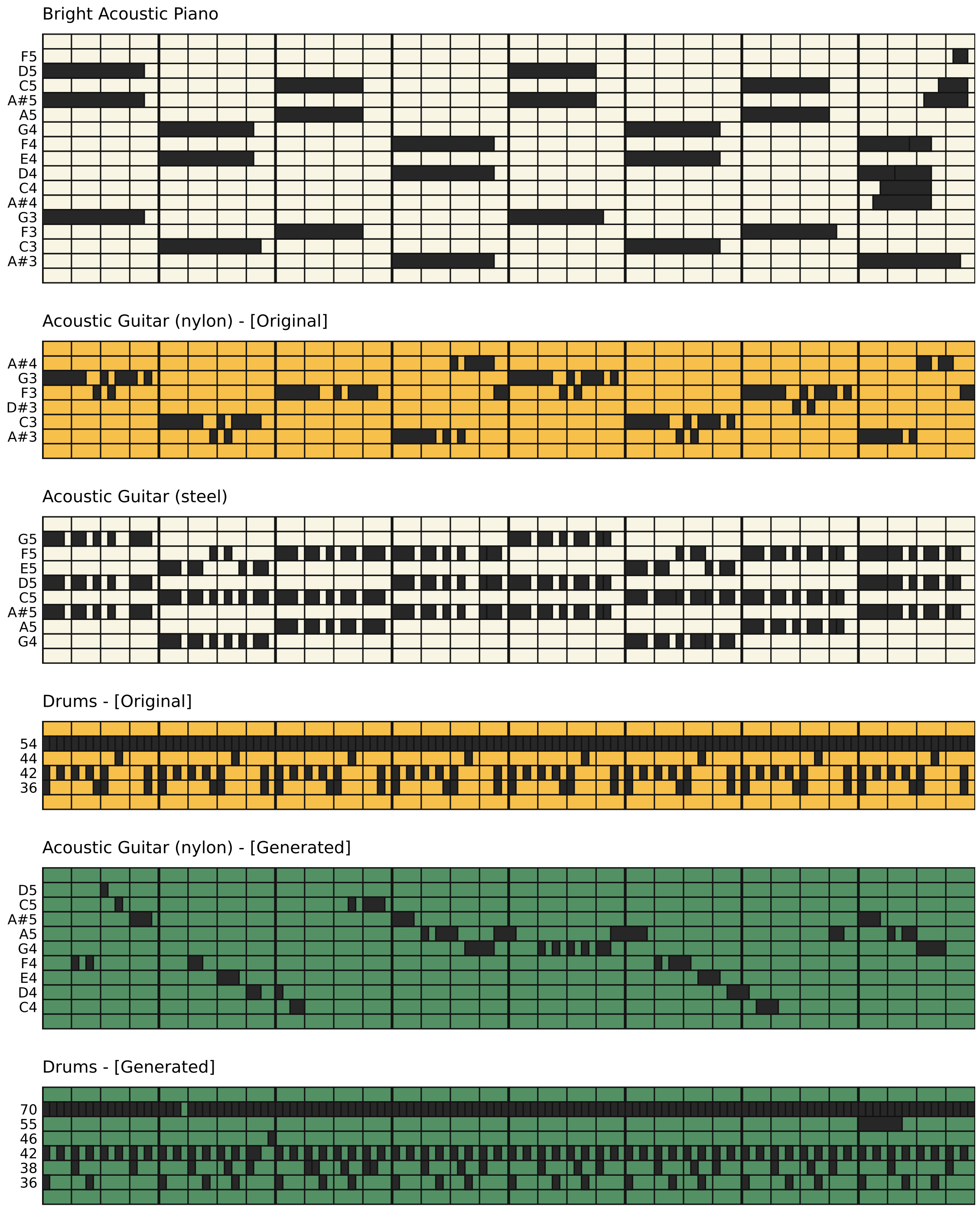


Figure F.14: An example of track infilling using MMM. The generation of each green track is conditioned on the beige tracks, replacing the original track(s) (shown in yellow).

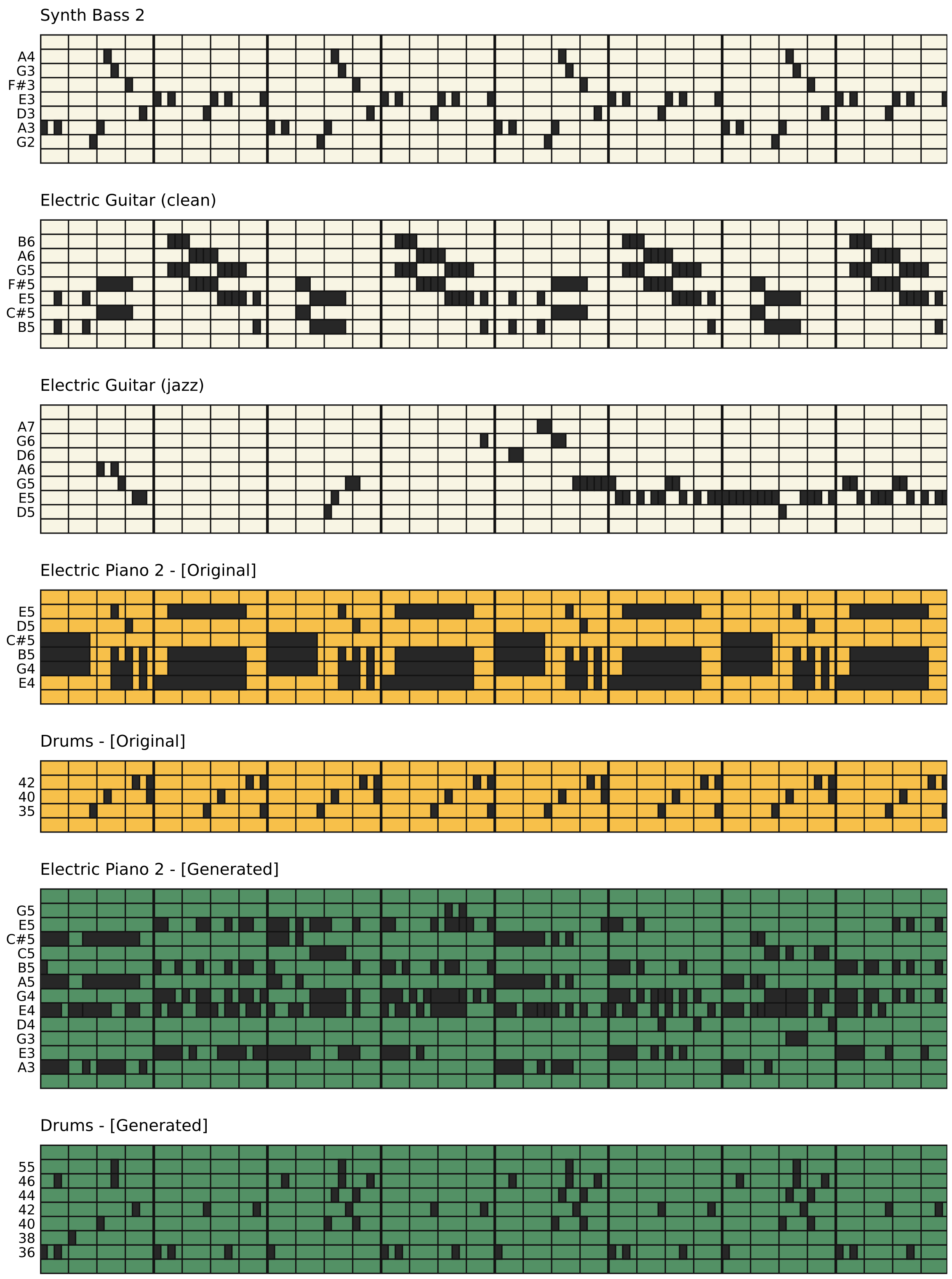


Figure F.15: An example of track infilling using MMM. The generation of each green track is conditioned on the beige tracks, replacing the original track(s) (shown in yellow).

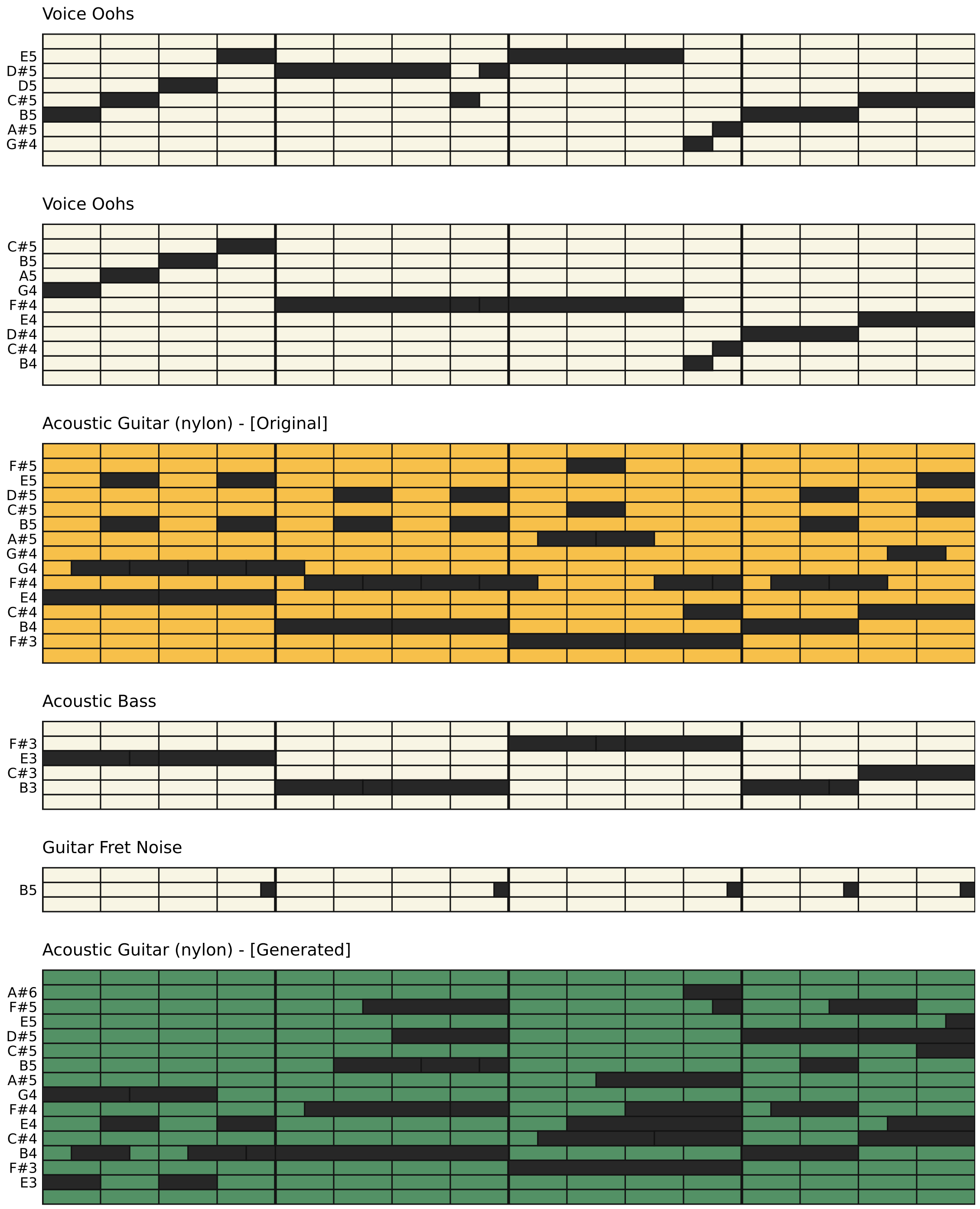


Figure F.16: An example of track infilling using MMM. The generation of each green track is conditioned on the beige tracks, replacing the original track(s) (shown in yellow).

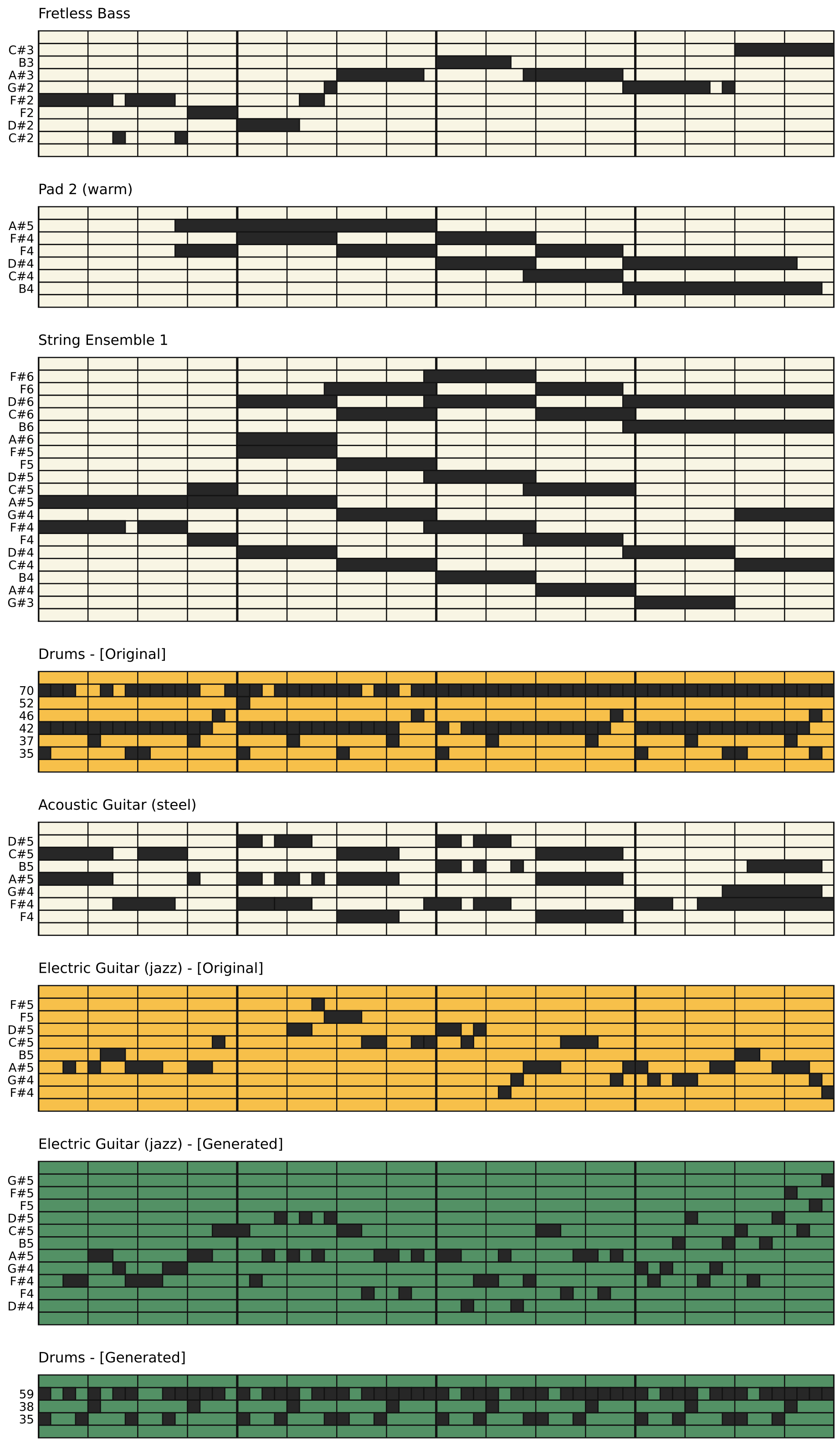


Figure F.17: An example of track infilling using MMM. The generation of each green track is conditioned on the beige tracks, replacing the original track(s) (shown in yellow).

F.3 Attribute Control - Note Density

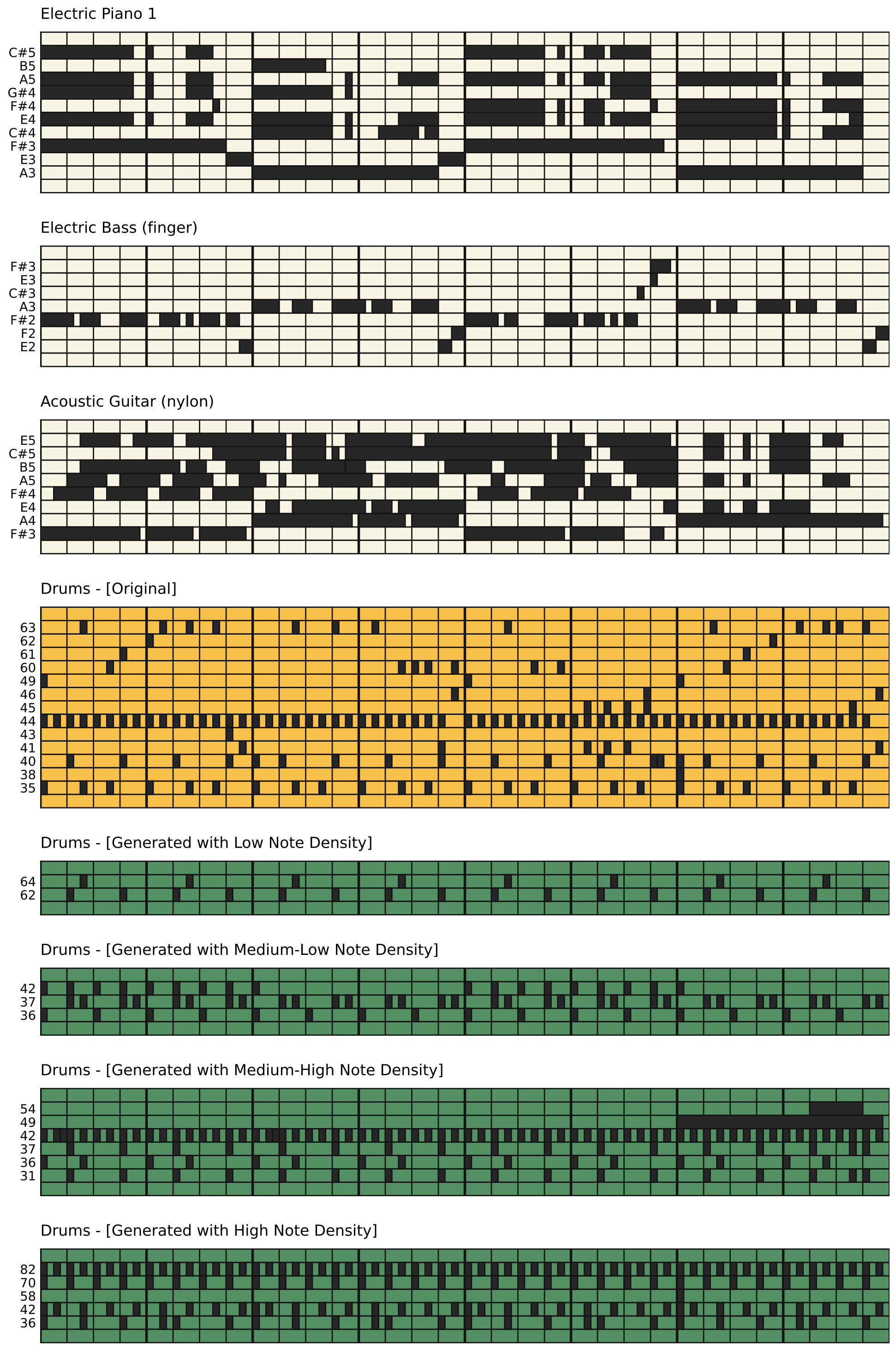


Figure F.18: An example of track infilling with note density attribute control using MMM. Each green track is conditioned on the beige tracks and a specific note density level.

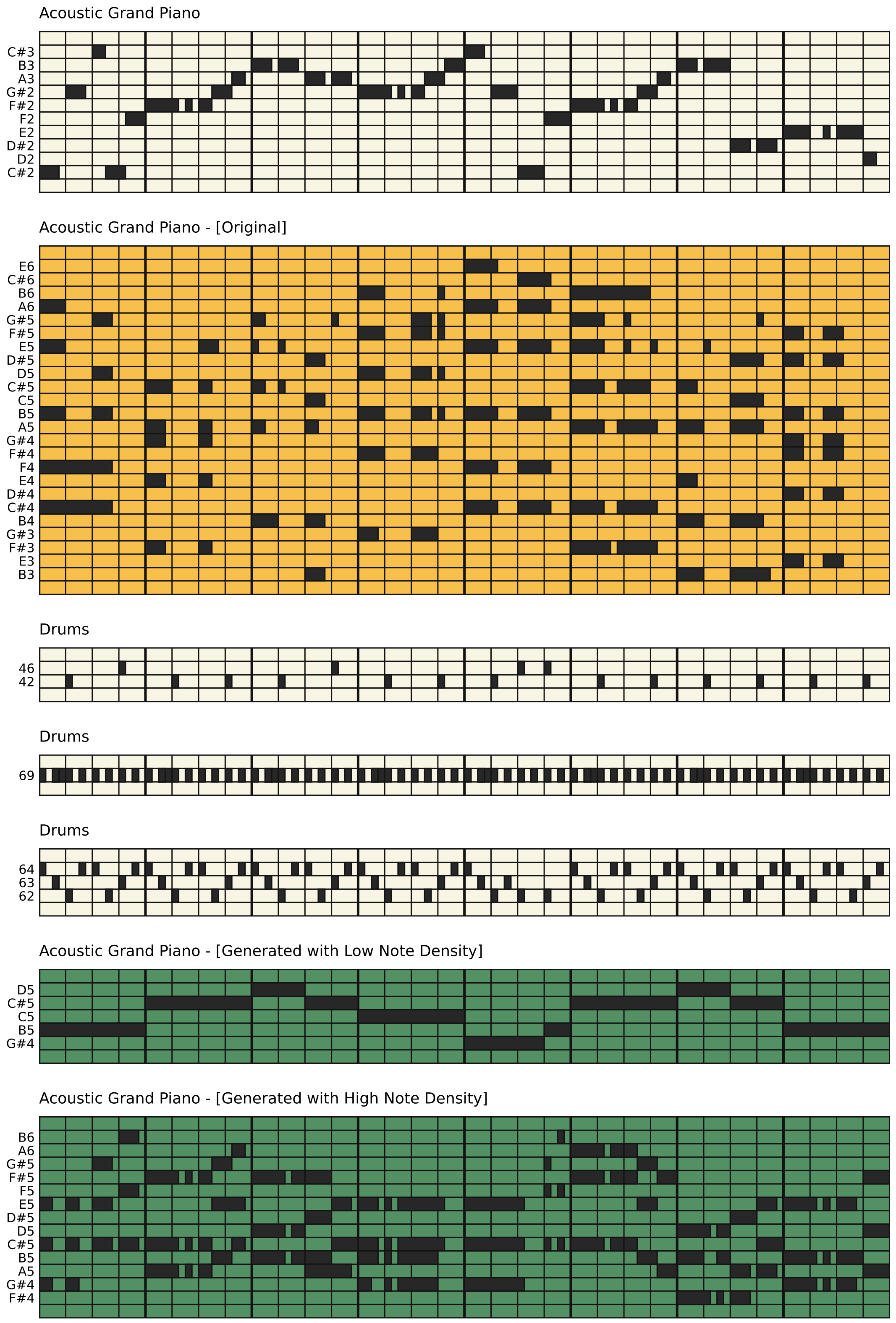


Figure F.19: An example of track infilling with note density attribute control using MMM. Each green track is conditioned on the beige tracks and a specific note density level.

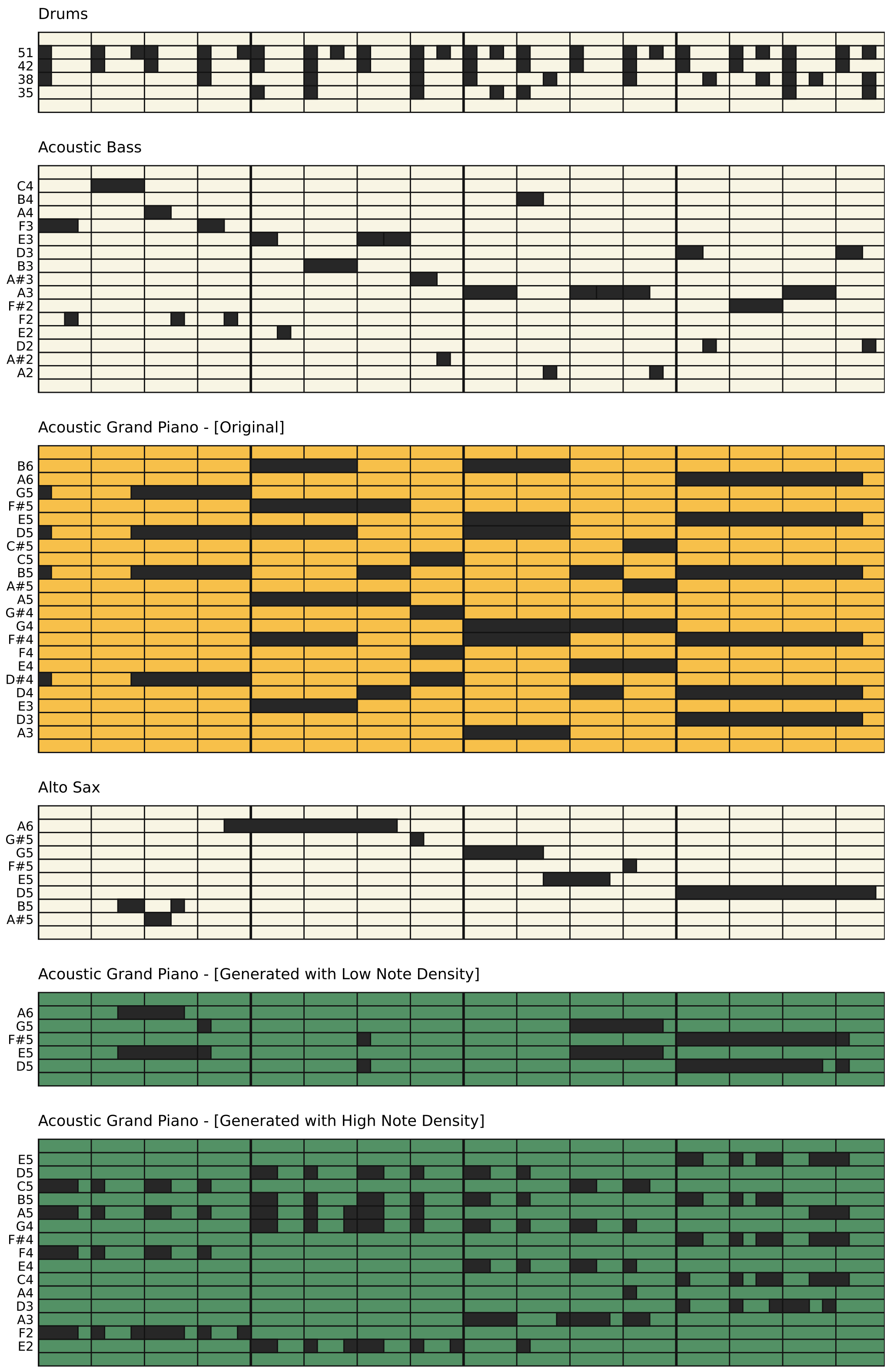


Figure F.20: An example of track infilling with note density attribute control using MMM. Each green track is conditioned on the beige tracks and a specific note density level.

F.4 Bar Infilling

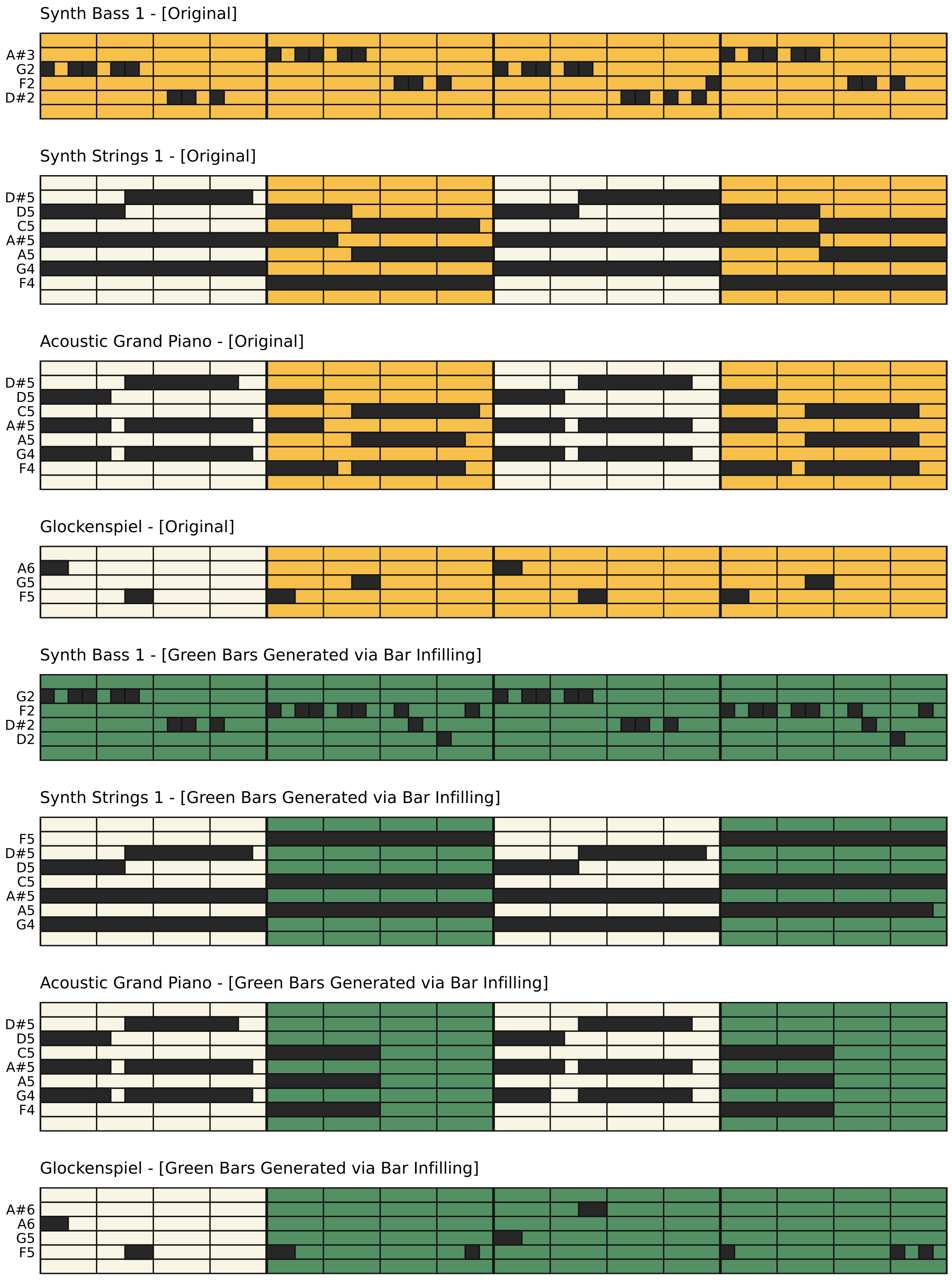


Figure F.21: An example of bar infilling using MMM. The yellow bars in the top four tracks are infilled, producing the green bars (conditioned on beige bars) shown in the bottom four tracks.